ESCOLA SUPERIOR DE TECNOLOGIA I CIÈNCIES EXPERIMENTALS

GRADO EN INGENIERÍA EN TECNOLOGÍAS INDUSTRIALES

Aplicación de la placa FPGA IceZUM Alhambra en prácticas de laboratorio de automatización de sistemas.

TRABAJO FIN DE GRADO

AUTOR

Andrei Balean

DIRECTOR

Julio Ariel Romero Pérez

Castellón, 10 de septiembre de 2018

Índice

Índice de figuras

Índice de tablas

1. Introducción

1.1 Antecedentes

El uso de mini-ordenadores de bajo costo, como Arduino o Raspberry-PI, es cada vez más común en la docencia universitaria. El trabajo con estos dispositivos requiere tiempos de aprendizaje cortos, lo cual permiten que los estudiantes se centren en conceptos básicos de programación y funcionamiento de los sistemas de control en lugar de perderse en aprender la configuración de un micro-controlador, más complejo y con una curva de aprendizaje mucho más costosa.

Actualmente se está desarrollando una iniciativa que sigue el "espíritu Arduino" en cuanto a facilidad de uso, pero que está enfocada en las FPGAs. Se trata de la placa IceZUM Alhambra, desarrollada por BQlabs, y el software para su programación llamado Icestudio. El trabajo con FPGAs profesionales requiere un conocimiento amplio de electrónica y de lenguajes de descripción de hardaware como Verilog o VHDL. El uso de Icestudio, sin embargo, simplifica el trabajo de forma considerable, permitiendo la programación gráfica basada en la conexión de puertas lógicas, biestables y otros elementos de la electrónica digital.

El objetivo del TFG es estudiar la viabilidad de este software y hardware para su uso en prácticas de laboratorio relacionadas con el diseño e implementación de circuitos digitales combinacionales y secuenciales. Con este objetivo se plantea estudiar la aplicación de dicha placa en el control de sistemas de eventos discretos simples. Para ello será necesario desarrollar las siguientes tareas:

- Diseñar la placa electrónica que sirva de interfaz entre la IceZUM Alhambra y los sensores y actuadores presentes en los sistemas a controlar.
- Plantear algunos problemas de control secuencial en los que sea factible el uso de la placa, teniendo en cuenta el número de entradas y salidas y la complejidad de los diseños obtenidos.
- Realizar el diseño de los circuitos combinaciones y/o secuenciales que realizan el control de acuerdo con las especificaciones dadas.
- Implementar los circuitos obtenidos en la placa IceZUM Alhambra, realizar la integración con los sistemas a controlar y comprobar su funcionamiento.

1.2 ¿Qué son las FPGAs?

Una **FPGA** o **matriz de puertas programables** (del inglés *Field Programmable Gate Array*) es un dispositivo programable que contiene bloques de lógica cuya interconexión y funcionalidad puede ser configurada in situ mediante un lenguaje de descripción especializado. La lógica programable puede reproducir desde funciones tan sencillas como las llevadas a cabo por una puerta lógica o un sistema combinacional hasta complejos sistemas en un chip.

Internamente están compuestos principalmente de cables, puertas lógicas, biestables, y puertos de entrada y salida. Todo ello sin conectar, como una plantilla en blanco, hasta que se les carga un archivo *bitstream* (un archivo generado a partir de la descripción del circuito).

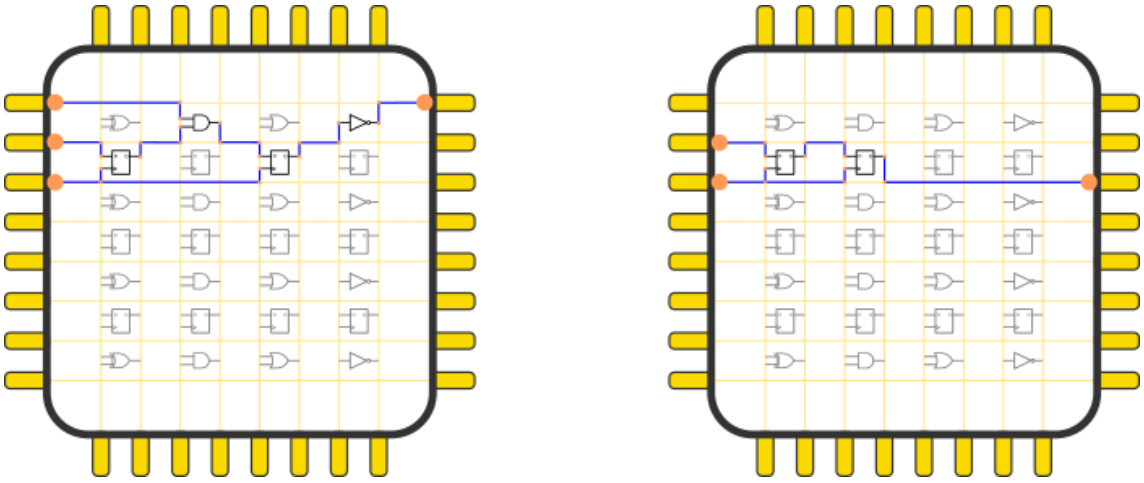


Figura 1: Estructura interna de un FPGA

La principal diferencia entre las FPGA y los microprocesadores es la complejidad. Aunque ambos varían en complejidad dependiendo de la escala, los microprocesadores tienden a ser más complejos que las FPGA. Esto se debe a los diversos procesos ya implementados en él.

Los microprocesadores ya tienen un conjunto fijo de instrucciones, que los programadores deben aprender para crear el programa de trabajo apropiado. Cada una de estas instrucciones tiene su propio bloque correspondiente que ya está cableado en el microprocesador. En cambio, una FPGA no tiene ningún bloque lógico cableado porque eso vencería el aspecto programable de campo del mismo.

Una FPGA se presenta como una red con cada unión que contiene un interruptor que el usuario puede hacer o romper. Esto determina cómo se determina la lógica de cada bloque.

1.3 FPGAs libres

Hasta hace poco tiempo, las FPGAs eran privativas, solo las grandes compañías sabían cómo funciona su interior. Esto era una desventaja porque creaba una barrera al conocimiento y no permitían avanzar rápidamente, conocer lo que hay allí dentro y modificarlo para cubrir unas necesidades concretas.

A pesar de que estos dispositivos llevan más de 30 años en el mercado, empiezan a ponerse de moda ahora y esto es debido a dos motivos. Por una parte, debido al progresivo estancamiento de la ley de Moore, las empresas ya no pueden delegar la mejora de la velocidad de sus algoritmos al continuo incremento de potencia de los procesadores.

Además, a finales de mayo del 2015 ocurrió un hecho significativo: es la primera vez que se tienen herramientas libres para realizar el ciclo completo de trabajo con FPGA usando únicamente herramientas libres. Esto es muy importante porque ya no se depende de ningún fabricante y todo el conocimiento está disponible para cualquiera que esté interesado. A partir de estas herramientas podemos crear cualquier interfaz, o cualquier aplicación que no haya sido prevista por el fabricante.

Estos hechos relevantes fueron posible gracias al proyecto se denominado "*Proyecto IceStorm*", creado por Clifford Wolf. El objetivo fue hacer ingeniería inversa de las FPGAs de Lattice para descifrar el formato del bitstream y poder crear herramientas libres. Nunca hasta ese momento se había podido lograr esto sin depender de las herramientas privativas, y muchas veces prohibitivas en cuanto a coste, del fabricante. Gracias a eso, se ha abierto el campo de las FPGAs para que más personas se introduzcan en un campo hasta ahora limitado principalmente a investigadores e industrias muy específicas.

2. IceZUM Alhambra

2.1 Introducción

Actualmente se puede distinguir entre dos tipos de FPGAs, libres y no libres. Las FPGAs libres son aquellos dispositivos que están equipados con los chips FPGA de la marca Lattice y son hardware de código abierto, es decir, cualquier persona puede disponer de todas las especificaciones y de los diagramas esquemáticos. Pero, además, estos dispositivos también se pueden programar utilizando software libre (de código abierto).

Por otro lado, las FPGAs no libres son aquellos dispositivos equipados con tecnología FPGA también, pero fabricados por empresas, las cuales son propietarias de las patentes de estos dispositivos y nadie puede acceder a ellas o copiarlas sin su permiso. Asimismo, estas empresas desarrollan también el único software capaz de programar los dispositivos fabricados por ellos.

Sin embargo, la placa utilizada para este proyecto es una FPGA libre, al igual que el software utilizado para su programación y puesta en funcionamiento. Gracias al software y hardware que han diseñado el equipo de desarrolladores de la IceZUM Alhambra, la programación y utilización de estos dispositivos son muy intuitivas y esto supone una ventaja en cuanto a los tiempos de aprendizaje.

2.2 Estudio de placas FPGA

Previamente se realizó un breve estudio sobre las placas FPGA libres disponibles en el mercado para poder compararlas y de esta manera elegir la mejor opción, en relación a las necesidades del proyecto.

Hay que especificar que el mercado de los dispositivos FPGA libres no es muy extendido debido a su reciente comienzo, pero, a pesar de ello, cada vez están surgiendo nuevos modelos. Además, se debe tener en cuenta que los modelos de placas FPGA libres están dotadas con el chip de un mismo fabricante, ya que el software libre existente solamente funciona para las placas que contienen los chips de la serie *iCE40HX o iCE40LP* fabricados por la empresa Lattice. Por tanto, las FPGAs libres, más destacadas, disponibles en el mercado, se muestran en la *Tabla 1*.

PLACA:	Icestick	Breakout Board	IceZUM Alhambra	Kéfir I
Referencia	ICE40HX1K-S-ENV	ICE40HX8K-B-ENV	-	-
Memoria Flash (Mbits)	32	32	32	32
Oscilador (MHz)	12	12	12	12
Conectores I/O (3,3V)	16	40	8	8
Conectores I/O (5V)	-	-	20	-
LEDs de usuario	5	8	8	4
Pulsador RESET	NO	NO	SI	NO
Interruptor ON/OFF	NO	NO	SI	NO
Pulsadores de usuario	-	-	2	4
Precio	25.25 €	40.90 €	65.00 €	No comercial
CHIP:	**iCE40HX1K**	**iCE40HX8K**	**iCE40HX1K**	**iCE40HX4K**
Elementos lógicos	1280	7680	1280	3520
Memoria (Kbits)	64	128	64	80
Pines FPGA de I/O	95	206	95	95

Tabla 1: Comparación de características de las placas FPGA

Como se puede observar no hay mucha variedad en el mercado, a pesar de esto, algunos modelos se descartaron directamente por no cumplir con los requisitos mínimos del proyecto. Además, de las placas mencionadas en la *Tabla 1*, también se tuvo que descartar la placa Kéfir I porque no se comercializa.

En cuanto a las dos primeras placas, son del mismo fabricante, Lattice, y son modelos comerciales, eso es un punto a favor, pero también un gran punto en contra es que no se diseñaron especialmente para la docencia, sino que para el mundo industrial y eso significa que para utilizarlas en este proyecto necesitarían varias modificaciones de hardware para facilitar su integración con los sistemas a controlar. Por tanto, se descartan también.

Finalmente se eligió la placa IceZUM Alhambra ya que fue diseñada y pensada para la docencia. A pesar de que el precio es algo más elevado, tiene varias características a favor como son, por ejemplo: los conectores de 5V, el pulsador de RESET, el interruptor ON/OFF, los LEDs de usuario y los pulsadores de usuario. Estas características permiten al usuario interactuar de manera fácil con la placa y de esta manera facilita el aprendizaje y comprensión del funcionamiento de esta tecnología.

2.3 La placa IceZUM Alhambra

La placa FPGA IceZUM Alhambra, desarrollada por BQlabs, tiene una apariencia muy similar a la placa Arduino UNO, de hecho, las dos tienen las mismas dimensiones y prácticamente los mismos pines (excepto alguna pequeña diferencia); la única gran diferencia entre estas dos placas es el funcionamiento de cada una. Fue diseñada así para que sea familiar a los usuarios y así facilitar el aprendizaje, y también para poder

utilizar la gran variedad de placas existentes que se utilizan como interfaces entre el Arduino UNO y otros circuitos externos, también conocidos como *Shields*. Se puede comparar y observar la similitud entre el Arduino UNO y la placa IceZUM Alhambra en la *Figura 2* y la *Figura 3,* expuestas a continuación.

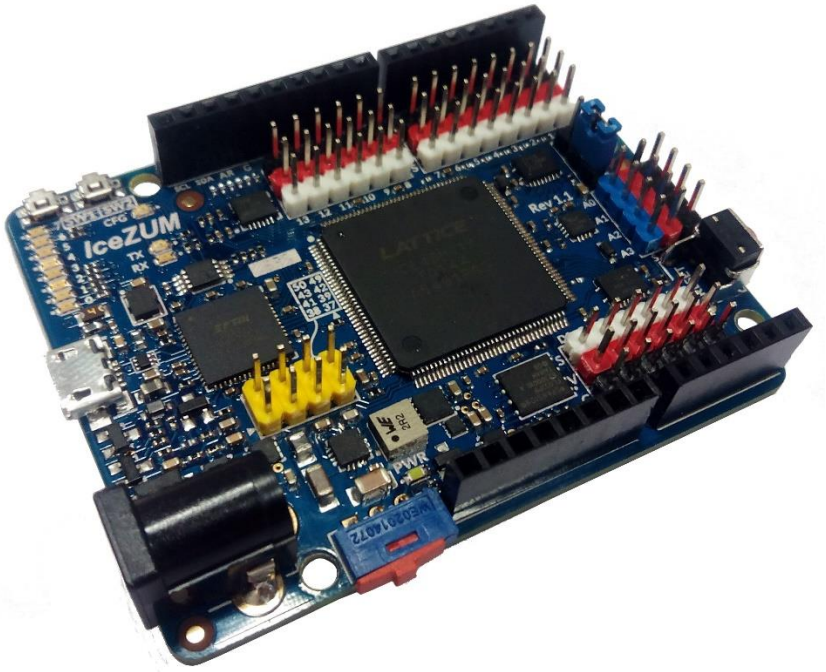


Figura 2: FPGA IceZUM Alhambra

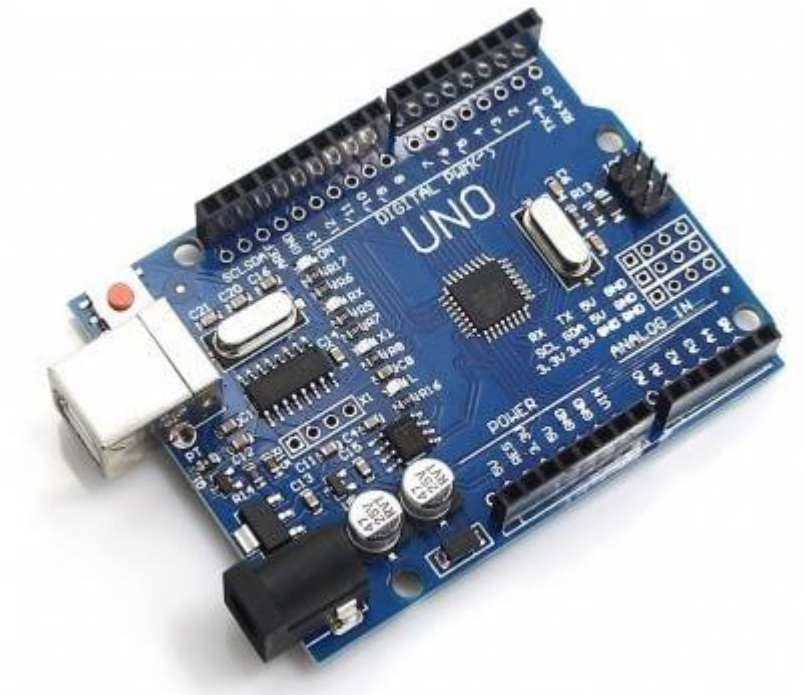


Figura 3: Arduino UNO

Las características más destacadas de la IceZUM Alhambra son las siguientes:

- Compatible con Icestorm/Icestudio
- Hardware libre
- Hasta 1280 elementos lógicos
- Oscilador de 12MHz
- Alimentación 6 - 17V
- 20 I/O pines a 5V
- 8 I/O pines a 3.3V
- Botón de RESET
- Botón de ON/OFF
- 8 LEDs de usuario
- 2 pulsadores de usuario
- Conector micro-USB tipo B
- Protección ante corto-circuitos y polaridad inversa

Una característica muy importante es que la placa dispone del conector micro-USB. Este se puede utilizar para conectar la placa con un ordenador y de esta manera poder programarla, pero además la placa puede ser alimentada mediante este conector, con una batería externa como si fuese un teléfono móvil, cosa que supone la posibilidad de utilizarla en una gran cantidad de aplicaciones como robots, coches teledirigidos o cualquier otra aplicación con necesidad de ser autónoma.

Otras facilidades que aporta la placa son los botones de RESET y ON/OFF, disponer de estas dos características es muy conveniente para aplicaciones como las mencionadas anteriormente.

2.4 Icestudio

Icestudio es el software desarrollado para programar la placa IceZUM Alhambra. Este programa está basado en Verilog, un lenguaje de descripción de hardware (HDL) aunque la interfaz del software con el usuario es totalmente gráfica, es decir, el programa muestra al usuario solamente compuertas lógicas, biestables y otros elementos lógicos básicos y, además, también dispone de bloques programables para los posibles usuarios más avanzados, que tengan conocimiento del lenguaje Verilog. Es por eso que facilita mucho la utilización.

En la *Figura 4* podemos observar como es la interfaz del software, algunos de sus elementos de diseño básicos y lo intuitiva que es.

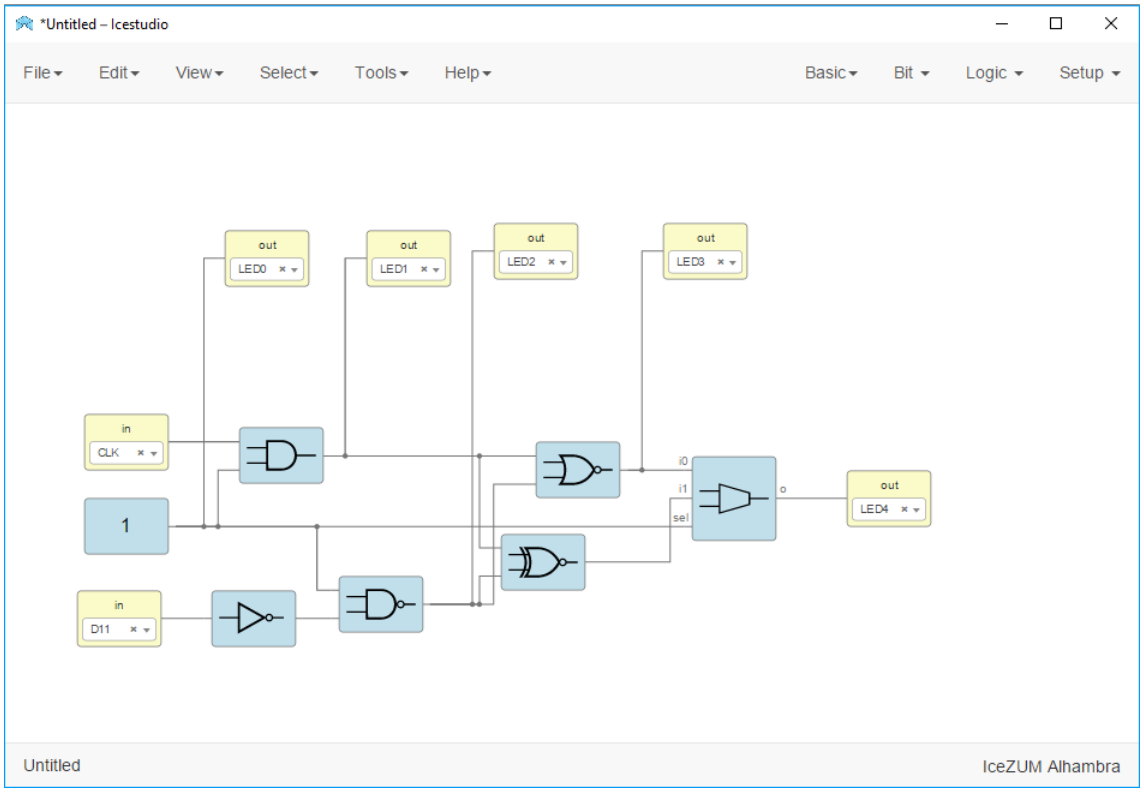


Figura 4: Interfaz de Icestudio

Aunque inicialmente el programa se desarrolló para la IceZUM Alhambra, en las versiones posteriores, lo adaptaron también para otras placas FPGA libres, todas ellas equipadas con chips Lattice de la serie iCE40HX. A continuación, se puede observar todas las placas soportadas por el software Icestudio.

Board name	GNU/Linux	Windows	Mac OS
IceZUM Alhambra	✓	✓	✓
Kéfir I iCE40-HX4K	✓	✓	✓
Nandland Go board	✓	✓	✓
iCE40-HX8K Breakout Board	✓	✓	✓
iCEstick Evaluation Kit	✓	✓	✓
icoBOARD 1.0	✓	✓	✓
BlackIce	✓	✓	✓
BlackIce II	✓	✓	✓
TinyFPGA B2	✓	✓	✓

Tabla 2: Placas soportadas por Icestudio.

3. PCB - Interfaz

3.1 Introducción

Para poder conectar a la placa que se está estudiando, sensores y actuadores necesarios para los ensayos prácticos, se ha diseñado una placa electrónica que hará de interfaz (*shield*, del inglés). Esta se conecta directamente a la placa IceZUM Alhambra.

El objetivo del *shield* es proteger la placa FPGA de las tensiones de funcionamiento de los sensores y actuadores industriales ya que estos funcionan a tensiones más elevadas (24 V) lo cual podría causar daños irreparables en la IceZUM Alhambra.

El diseño se empezó planteando una serie de necesidades para aprovechar al máximo todas las características de la placa FPGA sin superar unas dimensiones previamente establecidas. Por lo cual se decidió equipar el *shield* con 9 entradas y 7 salidas.

La placa electrónica se ha diseñado con el programa *EAGLE*, realizando primero el diseño de los esquemas de los circuitos electrónicos y seguidamente, ordenar todos los componentes de la mejor manera para luego dibujar las pistas de cobre.

3.2 Circuito de las entradas

Las 9 entradas, *IN1-IN9*, estarán conectadas en los pines 0-8 de la placa IceZUM Alhambra. El circuito de cada entrada está formado por un simple divisor de tensión.

Se le ha añadido un LED para poder comprobar visualmente que el circuito está funcionando y, además, para aportar más protección a la placa FPGA, se ha añadido un diodo zener de 5.1 voltios en paralelo a la resistencia R2. De esta manera nos aseguramos que en ningún momento habrá, en el pin de entrada, más tensión de la admitida. El circuito resultante se muestra a continuación, en la *Figura 5*.

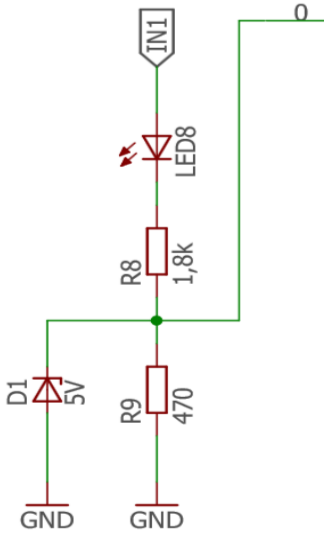


Figura 5: Esquema eléctrico del circuito de entrada

3.3 Circuito de las salidas

Las 7 salidas disponibles en el *shield*, OUT1-OUT7, están conectas a la placa IceZUM Alhambra a los pines 9 y D0-D5. El circuito de las salidas está compuesto por el circuito integrado ULN2003AN, formado por 7 transistores Darlington.

Además, se ha añadido a cada salida un LED para comprobar visualmente cuando el circuito está en funcionamiento. El esquema del circuito de las salidas es el siguiente:

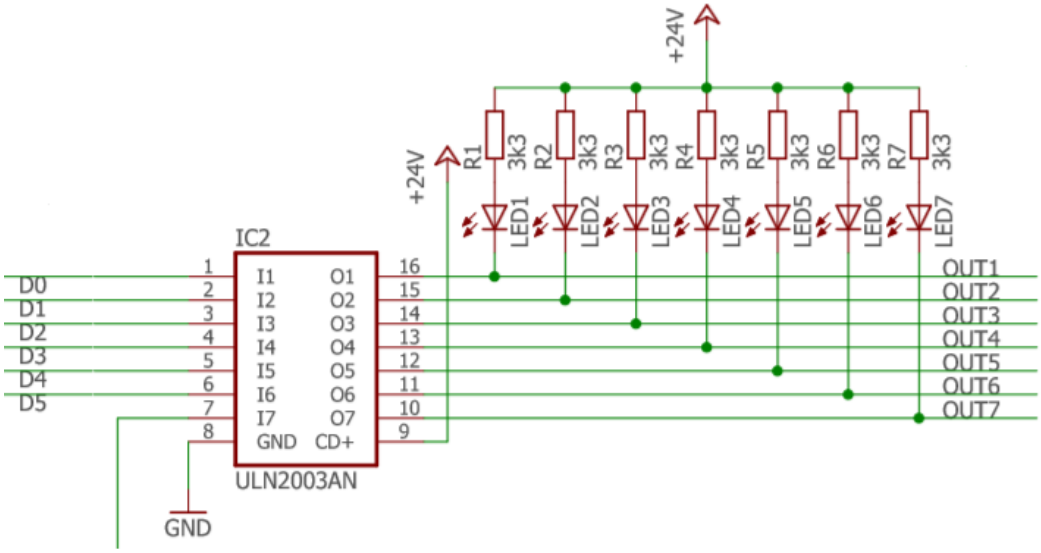


Figura 6: Esquema eléctrico del circuito de salida.

3.4 Regulador de tensión

En el diseño del *shield*, se ha equipado este con un regulador de voltaje del tipo LM7805. La función de este es regular una tensión de entrada (máximo 35V) para que la salida (VIN) proporcione 1A de corriente a 5V.

Gracias a la salida de 5V del regulador podremos alimentar también la placa IceZUM Alhambra, sin necesidad de aportar alimentación externa para esta. Por eso, el pin de salida del regulador se conectará directamente al pin de alimentación de la placa. En la *Figura 7* está representado el esquema del divisor de tensión.

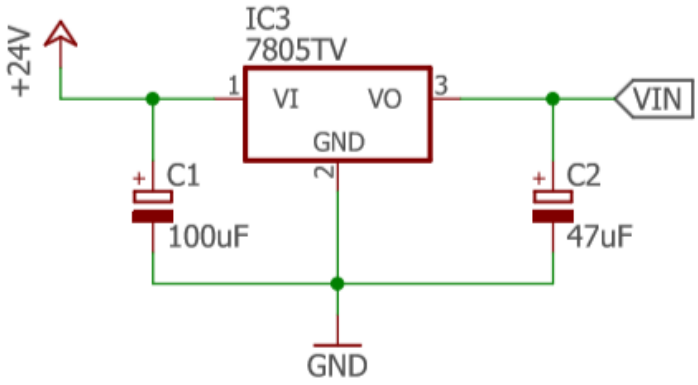


Figura 7: Esquema eléctrico del regulador de tensión

3.5 Diseño de la placa

Se ha tratado de ordenar los componentes electrónicos en la placa de la manera más simétrica y ordenada, teniendo en cuenta que los conectores de las entradas, las salidas y la alimentación deben estar al borde de la placa, para facilitar su conexión. Así mismo, también se ha tenido en cuenta la preferencia y/o necesidad de cercanía entre ciertos componentes. El esquemático completo del circuito impreso se puede consultar en el *Anexo 7.3*.

En las placas de circuito impreso (PCB, del inglés), a la hora de trazar las pistas, se recomienda utilizar solamente ángulos de 45 grados por distintas razones:

- Los ángulos de 90 grados producen puntas que pueden generar o emitir interferencias para señales de alta frecuencia.
- Los ángulos de 90 grados suelen ser más difíciles de mantener por las puntas en el proceso de quemado con ácido, por lo que usualmente el ancho de la pista se ve afectado lo que afecta la impedancia de la misma.
- Las trazas con ángulos a 45 grados lucen más estéticas y visualmente al ser más ordenadas es más fácil encontrar errores.

- Para señales por encima de 1 GHz, la impedancia se ve afectada por el área efectiva que se genera en los ángulos de 90°.

En las figuras mostradas a continuación se puede observar el diseño final de la placa electrónica.

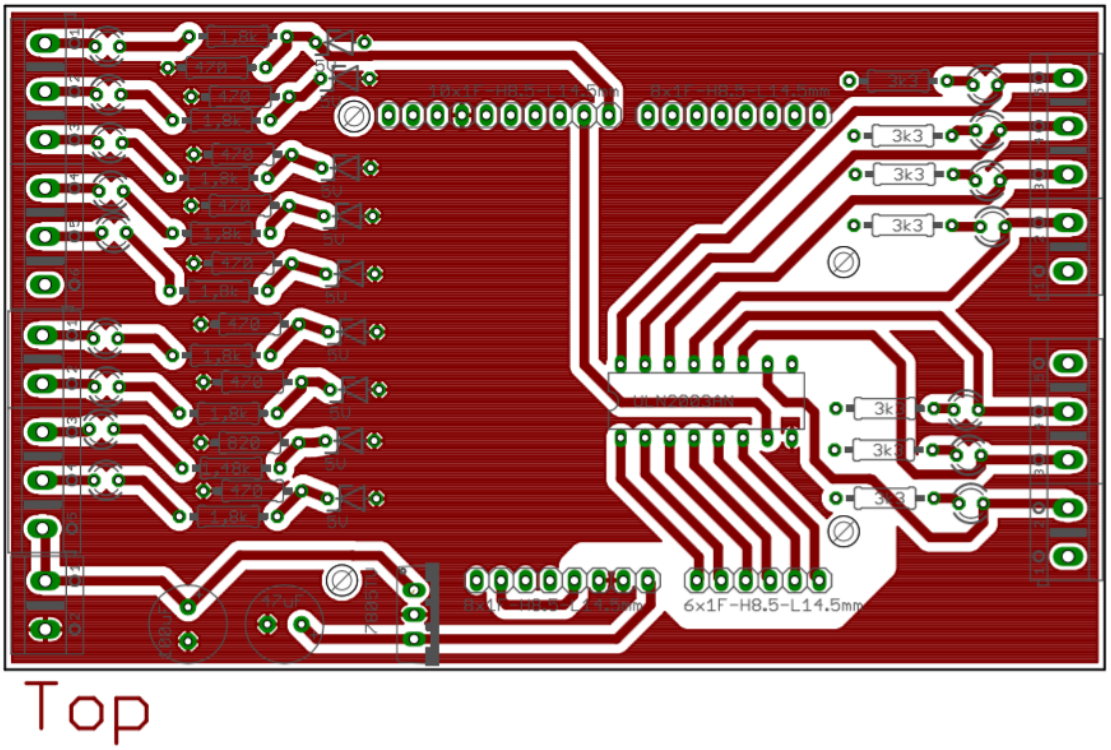


Figura 8: Esquema PCB – Parte superior

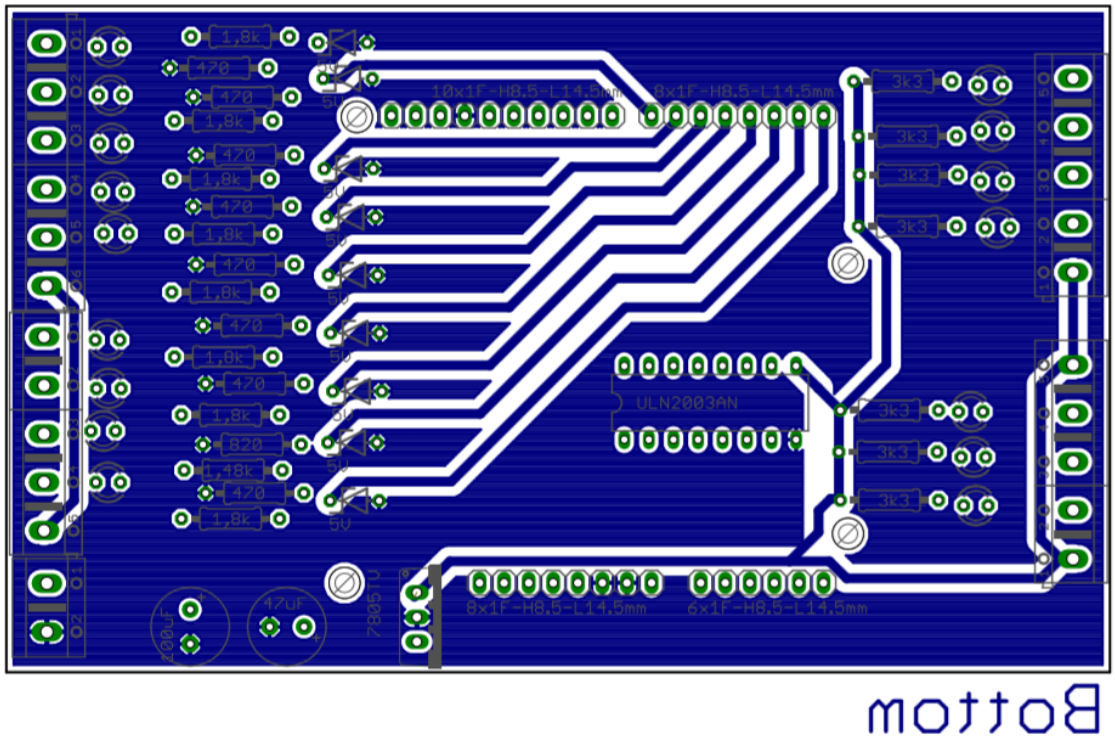


Figura 9: Esquema PCB – Parte inferior

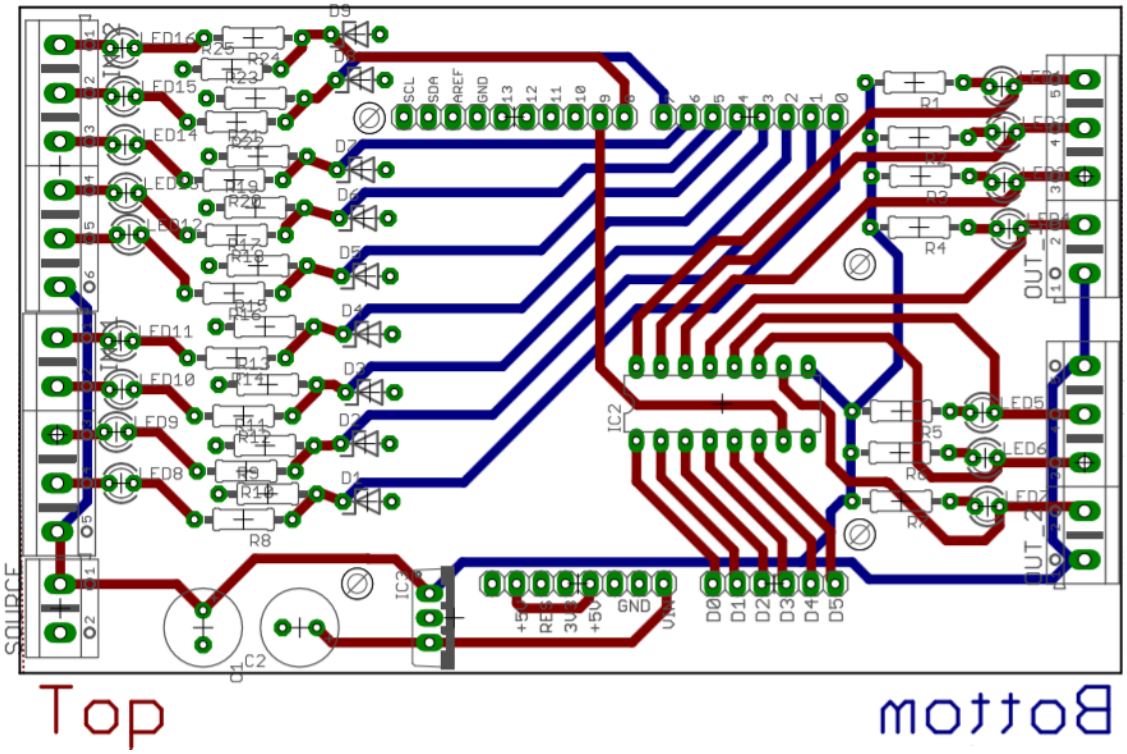


Figura 10: Esquema PCB – Ambas partes con componentes

3.6 Fabricación de la placa

La fabricación de la placa se realizó en el laboratorio de electrónica de la Universidad Jaume I y los resultados obtenidos fueron muy buenos, con una calidad del circuito impreso muy alta a pesar de la tecnología disponible utilizada. Asimismo, se utilizó una placa con doble capa de cobre, esto permitió dibujar pistas por la parte superior y también por la parte inferior, facilitando así el diseño.

En la figura mostrada a continuación, *Figura 11*, se observan los resultados obtenidos de la placa diseñada.

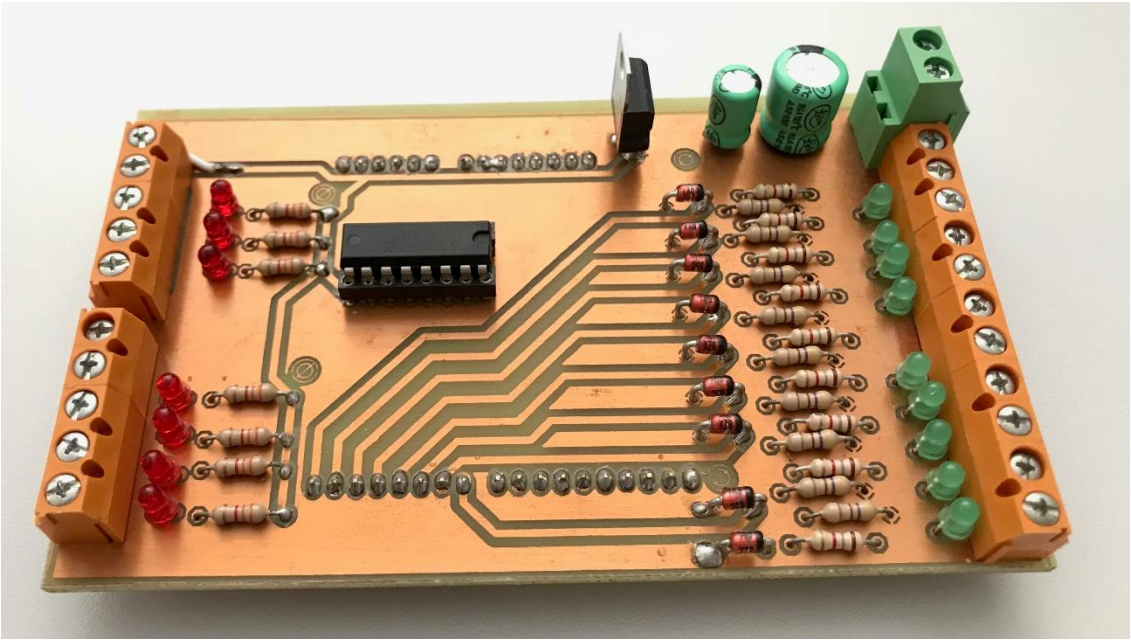


Figura 11: PCB - Interfaz

4. Programación de la FPGA

4.1 Implementación de circuitos combinacionales y secuenciales

En general para programar la FPGA se utiliza un lenguaje de programación especial, pero el software utilizado para la programación de la placa IceZUM Alhambra, Icestudio, permite hacerlo de dos maneras.

La primera consta en hacerlo directamente mediante funciones lógicas implementadas con compuertas lógicas (AND, OR, NAND, NOR, NOT, XNOR, XOR), células elementales de memoria (biestables) y/o multiplexores. Este método es bastante sencillo, el programador debe aprender o comprender lo que es la algebra booleana y seguir unos sencillos pasos dependiendo del problema que se presenta.

Por tanto, en función del tipo de circuito a diseñar y si este necesita memoria, se pueden distinguir dos tipos de circuito: los circuitos combinacionales y los circuitos secuenciales.

4.2 Implementación de circuitos mediante lenguaje HDL

Un lenguaje de descripción de hardware o *HDL* (*hardware description language,* del inglés) es un lenguaje de programación especializado que se utiliza para definir la estructura, diseño y operación de circuitos electrónicos.

Los lenguajes de descripción de hardware más utilizados son VHDL y Verilog, se puede decir que, en cierta forma, estos se parecen bastante a otros lenguajes de programación de ordenadores tales como el C o Java; básicamente consisten en una descripción textual con expresiones, declaraciones y estructuras de control. Sin embargo, una importante diferencia entre los HDL y otros lenguajes de programación está en que el HDL incluye explícitamente la noción de tiempo, debido a que este es una característica fundamental en los circuitos electrónicos reales.

En efecto, los HDL pueden ser usados para escribir especificaciones "ejecutables" de hardware. Es decir, un programa escrito en HDL hace posible que el diseñador de hardware pueda modelar y simular un componente electrónico antes de que este sea construido físicamente. Es esta posibilidad de "ejecución" de componentes lo que hace

que a veces los HDL se vean como lenguajes de programación convencionales, cuando en realidad se debería clasificarlos más precisamente como lenguajes de modelado.

El software Icestudio está basado en Verilog, por tanto, si el programador tiene conocimiento del lenguaje Verilog, existe también la posibilidad de programar la placa FPGA directamente en HDL.

En la *Figura 4.1*, expuesta a continuación, se muestra la programación de un multiplexor 2 a 1 en Verilog.

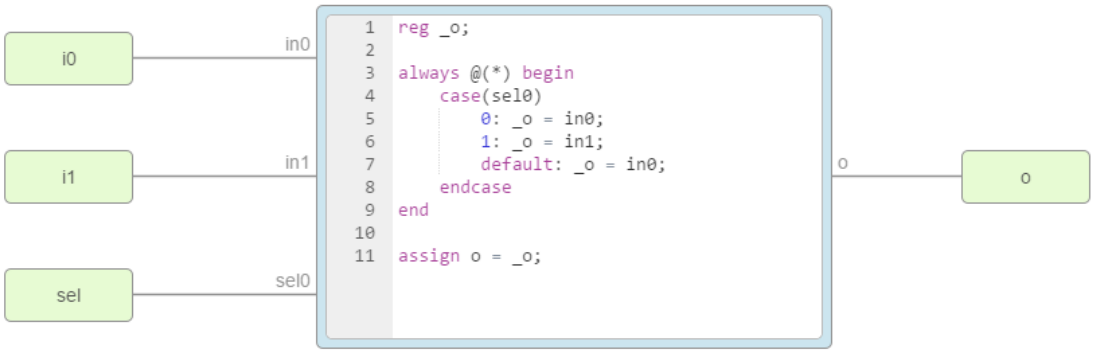


Figura 12: Multiplexor 2 a 1 en Verilog

5. Problemas propuestos

5.1 Introducción

En esta sección se propone una serie de problemas, con distinta dificultad, cuya implementación ha sido realizada mediante Icestudio. Los problemas seleccionados para este proyecto se han resuelto implementando sistemas combinacionales y sistemas secuenciales y se han ordenado en función de la dificultad de cada uno, empezando por el más sencillo y acabando con el más laborioso.

En la *Tabla 3* se ha esquematizado el orden de los problemas propuestos. Tal y como se puede observar en la tabla, la mayoría de los problemas se han dividido en varios bloques y se han resuelto parte por parte facilitando así su desarrollo.

Nº	Nombre	Nº de bloque	Tipo de resolución	Inputs	Biestables
1	*Limpia parabrisas*	1	Combinacional	3	0
		2	Secuencial	1	1
		Contador	Secuencial	1	2
2	*Ascensor de dos plantas*	1	Secuencial	4	2
3	*Ascensor de tres plantas*	1	Secuencial	2	1
		2	Secuencial	4	2
		3	Secuencial	2	1
4	*Manipulación con cilindros neumáticos*	1	Secuencial	4	2
		2	Secuencial	3	2
		3	Secuencial	3	2

Tabla 3: Clasificación de problemas

5.2 Problema 1. Limpia parabrisas

En este problema se desea programar el limpia parabrisas de un coche para que funcione a dos velocidades diferentes, determinadas por un interruptor externo, y además un modo corto y automático accionado al apretar un pulsador. Este limpia parabrisas se moverá gracias a un servomotor.

En el modo lento el servo permanecerá 2 segundos en cada posición y en el modo rápido medio segundo en cada posición. En cambio, al apretar el pulsador, el limpia parabrisas deberá funcionar igual que en el modo rápido, pero solamente se debe mover 3 veces y parar en la posición inicial.

El servomotor utilizado tendrá dos posiciones fijas, 0 y 1. La primera, 0, será cuando esté en reposo (con el limpia parabrisas recogido) y la segunda, 1, será cuando el limpia parabrisas estará abierto al máximo.

SOULCIÓN:

La programación para este problema se realiza mediante circuitos combinacionales y secuenciales y para facilitar el proceso se divide el problema en dos bloques.

5.2.1. Bloque 1 – Modo lento o rápido

En este bloque se diseña el programa que controlará los modos “lento” y “rápido” del limpia parabrisas, seleccionando el modo deseado mediante un interruptor externo.

Para este diseño se ha utilizado un multiplexor 2 a 1 y en cada entrada de este se ha conectado un generador de pulso (representado en Icestudio como un corazón) para que se active la salida en un cierto intervalo de tiempo. Para el modo rápido se ha utilizado una frecuencia de 1 Hz y para el modo lento se ha utilizado un pulso de 4 segundos el cuál es el equivalente a una frecuencia de 1/4 Hz. El pulso que genera la frecuencia de 1 Hz dura 1 segundo en un ciclo completo, eso quiere decir que el pulso estará a 1 durante medio segundo y a 0 durante otro medio segundo, por otro lado, el pulso generado por la frecuencia de 1/4 Hz estará en 1 durante 2 segundos y en 0 durante otros 2 segundos, por tanto, la duración total del pulso será de 4 segundos.

En la figura siguiente se muestra el diseño de este bloque y se puede observar la sencillez de este, aunque eso es debido al multiplexor que es un bloque e internamente es algo más complejo.

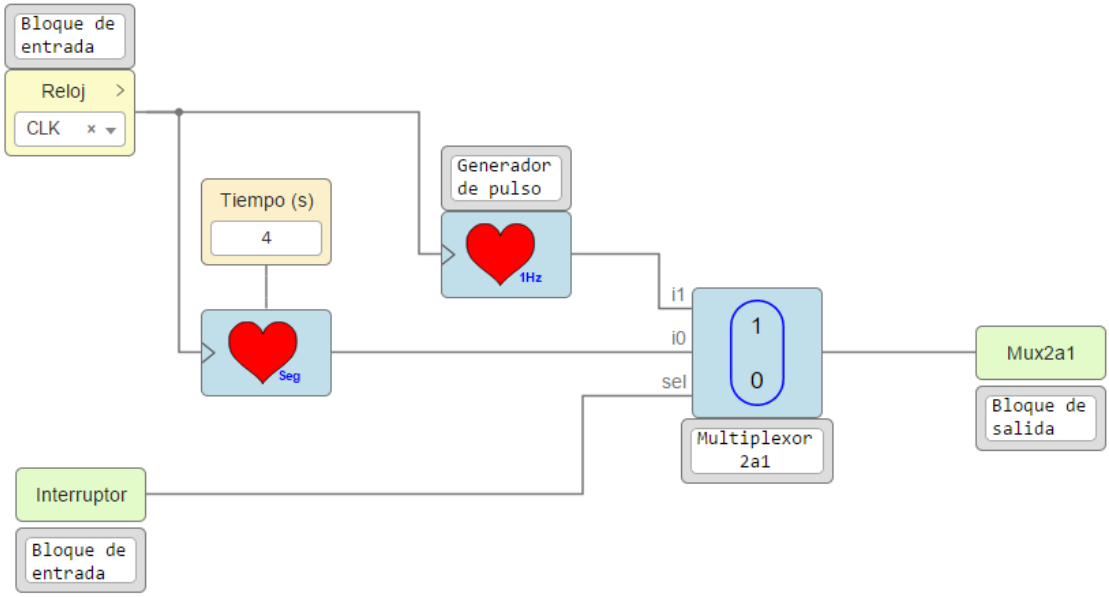


Figura 13: Modo lento-rápido

El multiplexor 2 a 1, como ya se ha mencionado anteriormente, mostrado como un bloque, se obtiene resolviendo un sistema combinacional. El propósito de este es, disponiendo de dos señales diferentes, proporcionar a la salida una de las señales disponibles en función de la elección del usuario o en función de la necesidad de otro bloque del problema a resolver.

La tabla de la verdad de este sistema combinacional es la siguiente:

Sel	A	B	S
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	1
1	1	1	1

Tabla 4: Estados posibles de un multiplexor 2 a 1

A partir de la tabla de la verdad y con la ayuda de los mapas de Karnaugh se ha obtenido la función lógica siguiente:

$$S = \mathrm{A} * \mathrm{Sel} + \mathrm{B} * \overline{\mathrm{Sel}}$$

Por último, se ha implementado el multiplexor, a partir de la expresión anterior, mediante puertas lógicas y su circuito se puede observar a continuación, en la *Figura 14*.

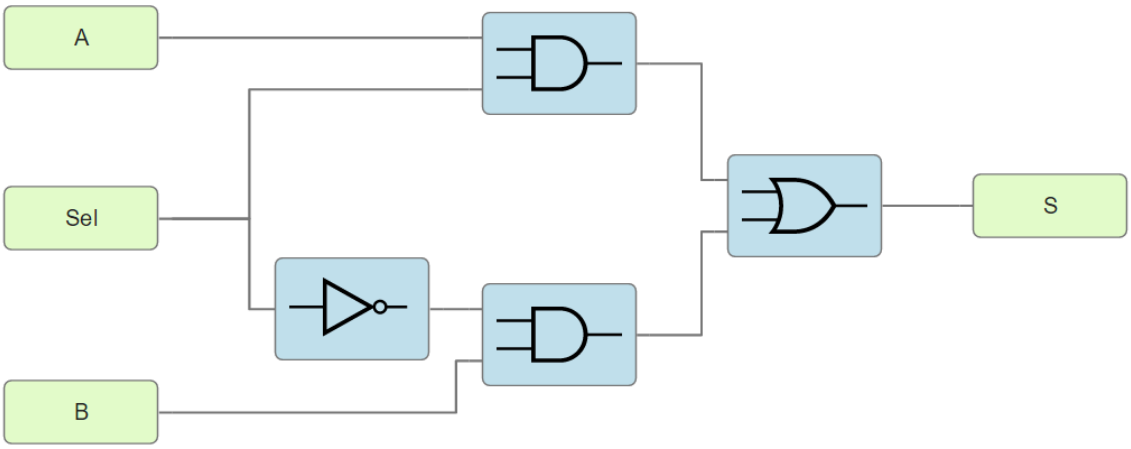


Figura 14: Multiplexor 2 a 1 con puertas lógicas

5.2.2. Bloque 2 – Modo automático

El diseño del modo automático, activado por el pulsador, está compuesto por un biestable de tipo RS, la salida del cual se debe activar al apretar el pulsador y se debe desactivar solamente cuando el limpia parabrisas se haya movido 3 veces. Para ello se necesita un contador de dos bits que desactive el modo automático, es decir, el biestable RS, cuando este llegue a su valor binario máximo 11 (equivalente al número 3 decimal). En la *Figura 15* se puede observar el diseño de un contador asíncrono de dos bits.

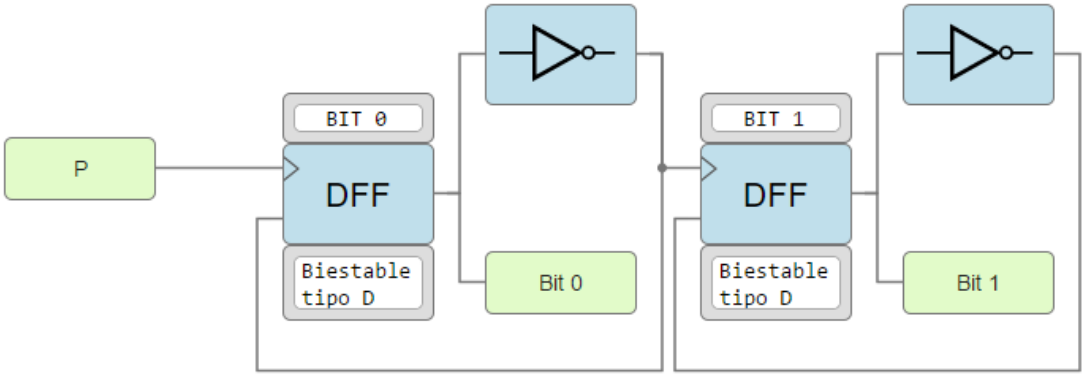


Figura 15: Contador de 2 bits

Por último, el pulso de este modo debe ser el mismo que en el modo “Rápido”, es decir, un pulso de 1 segundo. A continuación, se muestra el diseño completo de este

bloque en el cual se han utilizado el contador de dos bits para registrar las veces que el servo se moverá y de esta manera enviar una señal, cuando llegue a 3 (valor máximo), para resetear el biestable RS y de esta manera desactivar el circuito.

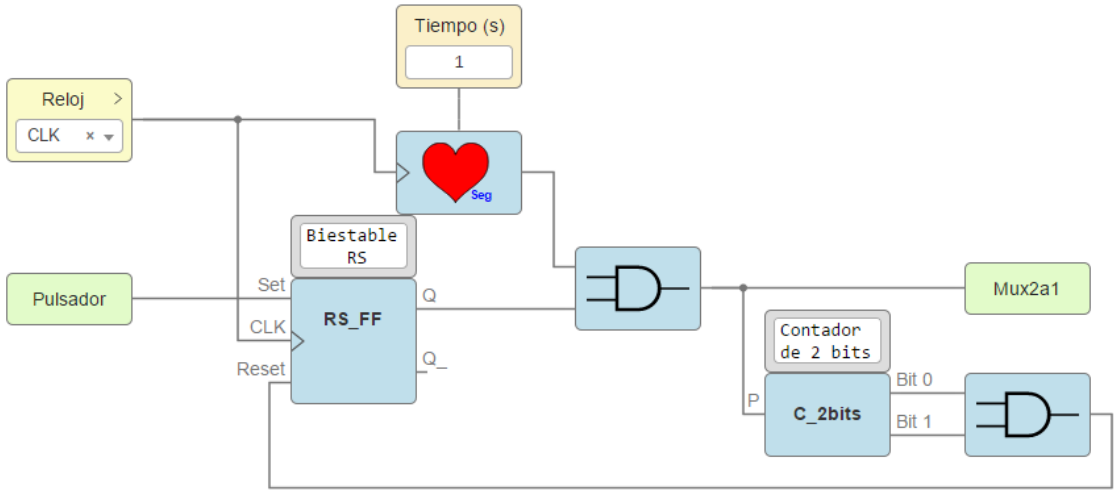


Figura 16: Modo automático

5.2.3. Solución final

Finalmente, en la solución completa del problema se han añadido los dos bloques previamente diseñados, cada uno conectados con su entrada digital necesaria. Adicionalmente, se ha añadido un multiplexor 2 a 1 para seleccionar cuál de los bloques controlará el servomotor, con la ayuda de otro interruptor externo. A continuación, se muestra la solución final implementada en Icestudio.

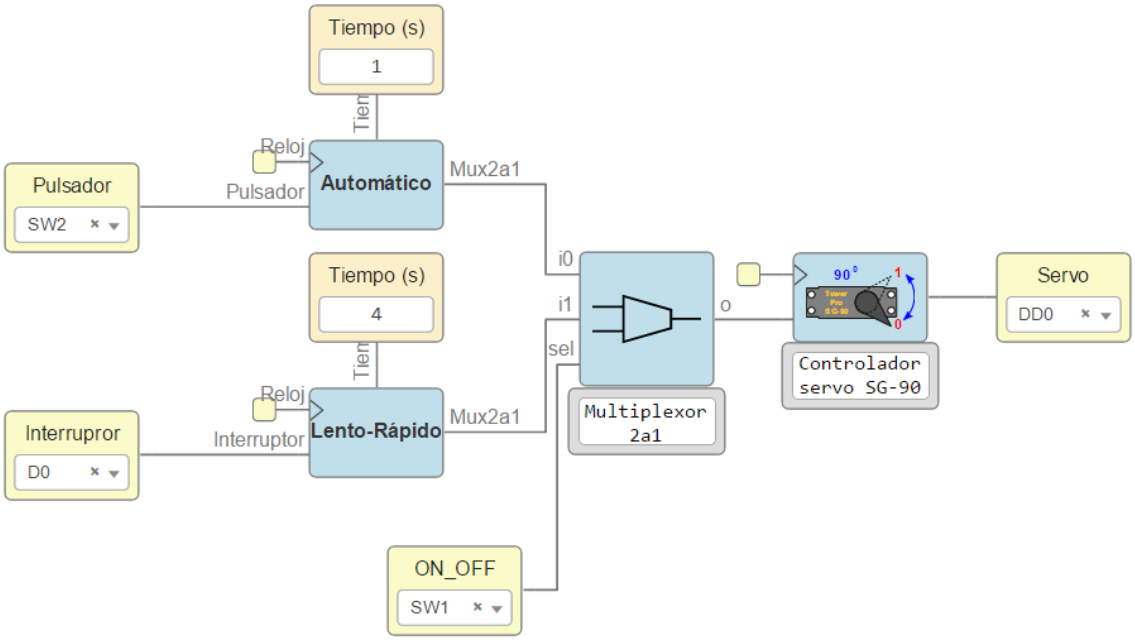


Figura 17: Solución final Problema 1

5.3 Problema 2. Ascensor de dos plantas

El problema propuesto consiste en el diseño de un automatismo que controle un ascensor de dos plantas. En cada una de las plantas habrá un sensor, **S1** y **S2**, que detecte si el ascensor se encuentra en dicha planta o no. Además, el automatismo dispone de dos pulsadores para poder mandar el ascensor a la planta deseada, **P1** para la primera planta y **P2** para la segunda.

Aunque en la vida real no funcionaría así, para facilitar la programación del problema, se va a prescindir de la puerta del ascensor. Por último, para mover el ascensor, el automatismo utilizará las señales **QS** y **QB** para activar el motor de este.

Las señales de entrada del automatismo serán:

- **P1** y **P2**, los botones del ascensor.
- **S1** y **S2**, los sensores, colocados en cada planta. Están activos si el ascensor está en esa planta.

Las señales de salida del automatismo serán:

- **QS**. mueve el ascensor hacia arriba.
- **QB**. Mueve el ascensor hacia abajo.

SOULCIÓN:

El automatismo de este problema tiene dos señales de salida (**QS** y **QB**), por lo que el ascensor tendrá tres estados posibles (*Subir, Bajar y Parado*), lo que significa que el diseño de su resolución contendrá dos biestables de tipo D, ya que se necesitan dos bits para poder expresar los tres estados en binario. A continuación, en la *Figura 5.4*, se muestra el diagrama de estados del ascensor.

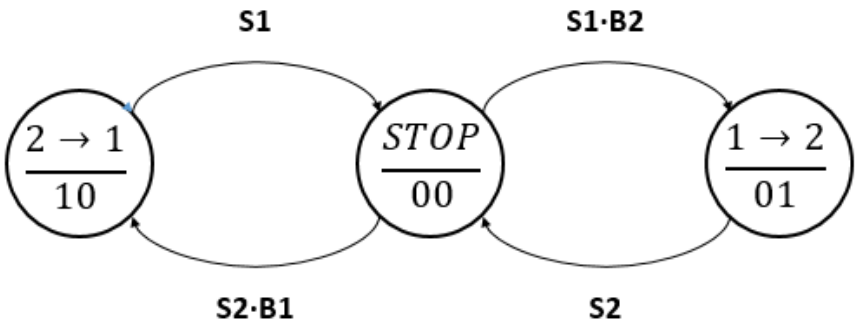


Figura 18: Diagrama de estados del ascensor de dos plantas

En la tabla siguiente se han expuesto todos los casos posibles en el funcionamiento del ascensor, en función de las entradas activas. De esta manera, a partir de la tabla de la verdad y mediante el método de Karnaugh, se obtendrán las ecuaciones para un funcionamiento correcto.

Estado actual		**INPUTS**				**Estado siguiente**		**Flip-Flop D**	
A	B	S1	S2	B1	B2	An	Bn	DA	DB
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	1	0	0	0	0
0	0	0	0	1	0	0	0	0	0
0	0	0	0	1	1	0	0	0	0
0	0	0	1	0	0	0	0	0	0
0	0	0	1	0	1	0	0	0	0
0	0	0	1	1	0	1	0	1	0
0	0	0	1	1	1	1	0	1	0
0	0	1	0	0	0	0	0	0	0
0	0	1	0	0	1	0	1	0	1
0	0	1	0	1	0	0	0	0	0
0	0	1	0	1	1	0	1	0	1
0	0	1	1	0	0	X	X	X	X
0	0	1	1	0	1	X	X	X	X
0	0	1	1	1	0	X	X	X	X
0	0	1	1	1	1	X	X	X	X
0	1	0	0	0	0	0	1	0	1
0	1	0	0	0	1	0	1	0	1
0	1	0	0	1	0	0	1	0	1
0	1	0	0	1	1	0	1	0	1
0	1	0	1	0	0	0	0	0	0
0	1	0	1	0	1	0	0	0	0
0	1	0	1	1	0	0	0	0	0
0	1	0	1	1	1	0	0	0	0
0	1	1	0	0	0	0	1	0	1
0	1	1	0	0	1	0	1	0	1
0	1	1	0	1	0	0	1	0	1
0	1	1	0	1	1	0	1	0	1
0	1	1	1	0	0	X	X	X	X
0	1	1	1	0	1	X	X	X	X
0	1	1	1	1	0	X	X	X	X
0	1	1	1	1	1	X	X	X	X
1	0	0	0	0	0	1	0	1	0
1	0	0	0	0	1	1	0	1	0
1	0	0	0	1	0	1	0	1	0
1	0	0	0	1	1	1	0	1	0

1	0	0	1	0	0	1	0	1	0
1	0	0	1	0	1	1	0	1	0
1	0	0	1	1	0	1	0	1	0
1	0	0	1	1	1	1	0	1	0
1	0	1	0	0	0	0	0	0	0
1	0	1	0	0	1	0	0	0	0
1	0	1	0	1	0	0	0	0	0
1	0	1	0	1	1	0	0	0	0
1	0	1	1	0	0	X	X	X	X
1	0	1	1	0	1	X	X	X	X
1	0	1	1	1	0	X	X	X	X
1	0	1	1	1	1	X	X	X	X
1	1	X	X	X	X	X	X	X	X

Tabla 5: Estados posibles del ascensor de dos plantas

A partir de la *Tabla 5* se han obtenido las ecuaciones necesarias para el funcionamiento correcto del ascensor y son las siguientes:

$$D_A = A * \overline{S1} + \overline{B} * S2 * B1$$

$$D_B = B * \overline{S2} + \overline{A} * S1 * B2$$

A continuación, a partir de las ecuaciones obtenidas, se puede diseñar la programación del problema implementando un sistema secuencial. Para esto se han utilizado las puertas lógicas necesarias, 4 entradas digitales y 2 salidas digitales. Además, por seguridad, se han restringido las salidas para que no puedan estar activas al mismo tiempo. En la *Figura 19* se observa el diseño final del problema, implementado en Icestudio.

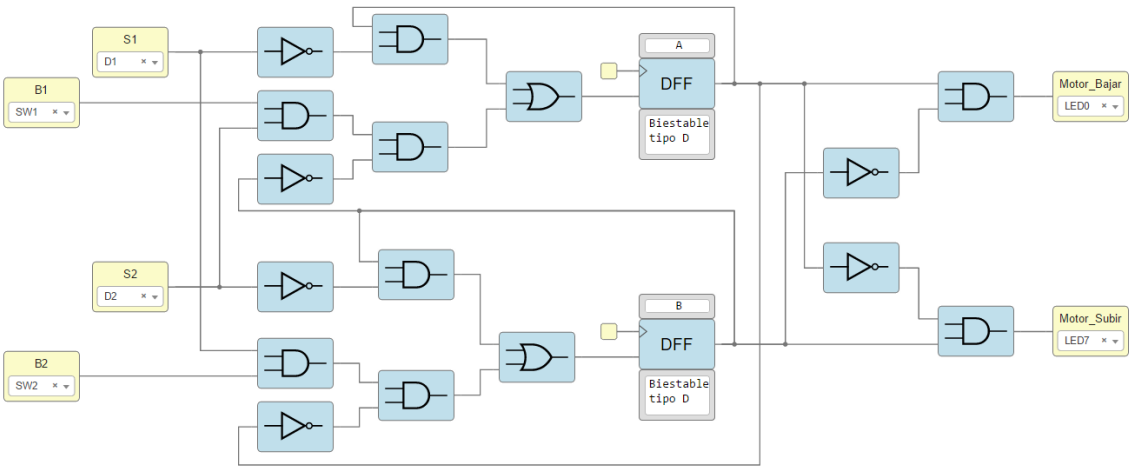


Figura 19: Diseño en Icestudio del ascensor de dos plantas

5.4 Problema 3. Ascensor de tres plantas

El siguiente problema consiste en el diseño de un automatismo que controle un ascensor de tres plantas. Al igual que en el problema anterior, para facilitar la programación del problema, se va a prescindir de la puerta del ascensor.

El funcionamiento supone que, estando el ascensor en reposo en cualquiera de las plantas, al apretarse alguno de los botones para ir a otra planta, este vaya a la planta correspondiente sin tener en cuenta que debería de estar cerrada la puerta antes de moverse.

Las señales de entrada del automatismo serán:

- **P1**, **P2** y **P3** son los botones del ascensor, uno para cada planta.
- **S1**, **S2** y **S3** son los sensores, colocados en cada planta. Están activos si el ascensor está en esa planta.

Las señales de salida del automatismo serán:

- **QS**. mueve el ascensor hacia arriba.
- **QB**. Mueve el ascensor hacia abajo.

SOLUCIÓN:

Al igual que en el problema anterior, el automatismo de este problema tiene dos señales de salida (**QS** y **QB**) pero, al tener una planta más, la resolución se dividirá en tres partes.

5.4.1 Bloque 1 – Funcionamiento al apretar P1

La primera parte consiste en cómo funcionará el ascensor cuando se apriete el pulsador P1. Eso significa que, estando el ascensor en la segunda o tercera planta, al apretarse P1 este debe bajar a la primera planta. Por tanto, en este caso, los únicos estados del ascensor será *Bajar* o *Parado*. En la *Figura 20* se muestra el diagrama de estados de esta primera parte.

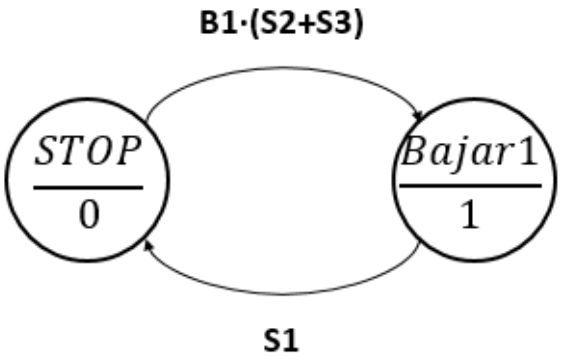


Figura 20: Diagrama de estados P1

Para la resolución de esta parte se necesitará un solo biestable D, ya que se puede determinar los dos estados con un solo bit.

En la tabla siguiente se han expuesto todos los casos posibles en el funcionamiento del ascensor en el momento en que se pulse P1, en función de las entradas activas en ese momento. De esta manera, a partir de la tabla y mediante el método de Karnaugh, se obtendrán las ecuaciones para un funcionamiento correcto.

Estado actual	**INPUTS**				**Estado siguiente**	**Flip-Flop D**
A	P1	S1	S2	S3	An	DA
0	0	0	0	0	0	0
0	0	0	0	1	0	0
0	0	0	1	0	0	0
0	0	0	1	1	0	0
0	0	1	0	0	0	0
0	0	1	0	1	0	0
0	0	1	1	0	0	0
0	0	1	1	1	0	0
0	1	0	0	0	1	1
0	1	0	0	1	1	1
0	1	0	1	0	1	1
0	1	0	1	1	1	1
0	1	1	0	0	0	0
0	1	1	0	1	X	X
0	1	1	1	0	X	X
0	1	1	1	1	X	X
1	0	0	0	0	1	1
1	0	0	0	1	1	1
1	0	0	1	0	1	1
1	0	0	1	1	1	1

1	0	1	0	0	0	0
1	0	1	0	1	0	0
1	0	1	1	0	0	0
1	0	1	1	1	0	0
1	1	0	0	0	1	1
1	1	0	0	1	1	1
1	1	0	1	0	1	1
1	1	0	1	1	1	1
1	1	1	0	0	0	0
1	1	1	0	1	0	0
1	1	1	1	0	0	0
1	1	1	1	1	0	0

Tabla 6: Estados posibles al apretar P1

A partir de la *Tabla 6* se obtienen las ecuaciones del bloque 1 y son las siguientes:

$$D_A = (P1 + A) * \overline{S1}$$

A partir de la ecuación obtenida se puede diseñar la programación del problema implementando un sencillo sistema secuencial, utilizando tan solo 3 puertas lógicas, 2 entradas digitales, una entrada digital y solamente un biestable, ya que el bloque tiene solamente una ecuación. En la *Figura 21* se puede observar el circuito obtenido.

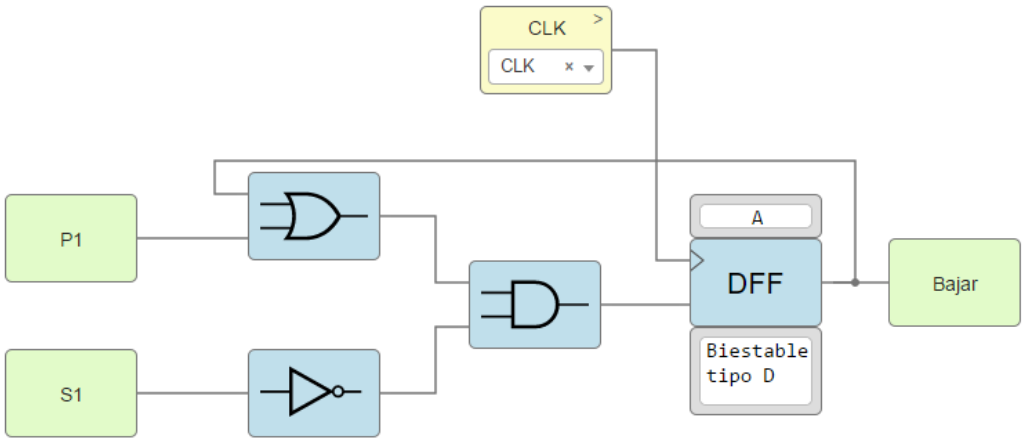


Figura 21: Diseño en Icestudio del funcionamiento apretando P1

5.4.2 Bloque 2 – Funcionamiento al apretar P2

La segunda parte consiste en cómo funcionará el ascensor cuando se apriete el pulsador P2. Eso significa que, estando el ascensor en la primera o tercera planta, al apretarse P2 este debe subir (o bajar) a la planta respectiva. Por tanto, en este caso, el ascensor podrá estar en tres estados, *Bajar*, *Subir* o *Parado*. En la *Figura 22* se muestra el diagrama de estados de este bloque.

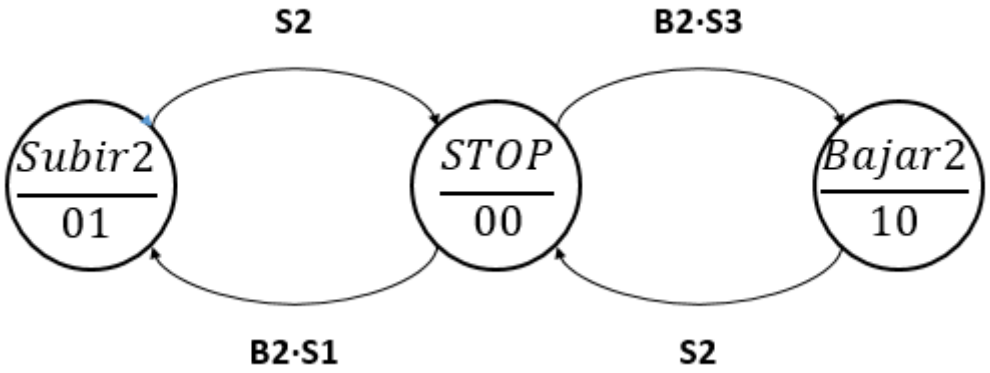


Figura 22: Diagrama de estados P2

Por el hecho de tener tres estados posibles, en la resolución de esta parte se van a necesitar dos biestables D, ya que para determinar los diferentes estados harán falta dos bits.

En la tabla siguiente se han expuesto todos los casos posibles en el funcionamiento del ascensor en el momento en que se apriete P2, en función de las entradas activas en ese momento. De esta manera, a partir de la tabla y mediante el método de Karnaugh, se obtendrán las ecuaciones para un funcionamiento correcto.

Estado actual			**INPUTS**			**Estado siguiente**		**Flip-Flop D**	
A	B	P2	S1	S2	S3	An	Bn	DA	DB
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	1	0	0	0	0
0	0	0	0	1	0	0	0	0	0
0	0	0	0	1	1	0	0	0	0
0	0	0	1	0	0	0	0	0	0
0	0	0	1	0	1	0	0	0	0
0	0	0	1	1	0	0	0	0	0
0	0	0	1	1	1	0	0	0	0

0	0	1	0	0	0	0	0	0	0
0	0	1	0	0	1	1	0	1	0
0	0	1	0	1	0	0	0	0	0
0	0	1	0	1	1	X	X	X	X
0	0	1	1	0	0	0	1	0	1
0	0	1	1	0	1	X	X	X	X
0	0	1	1	1	0	X	X	X	X
0	0	1	1	1	1	X	X	X	X
0	1	0	0	0	0	0	1	0	1
0	1	0	0	0	1	0	0	0	0
0	1	0	0	1	0	0	0	0	0
0	1	0	0	1	1	0	0	0	0
0	1	0	1	0	0	0	1	0	1
0	1	0	1	0	1	0	0	0	0
0	1	0	1	1	0	0	0	0	0
0	1	0	1	1	1	0	0	0	0
0	1	1	0	0	0	0	1	0	1
0	1	1	0	0	1	0	0	0	0
0	1	1	0	1	0	0	0	0	0
0	1	1	0	1	1	0	0	0	0
0	1	1	1	0	0	0	1	0	1
0	1	1	1	0	1	0	0	0	0
0	1	1	1	1	0	0	0	0	0
0	1	1	1	1	1	0	0	0	0
1	0	0	0	0	0	1	0	1	0
1	0	0	0	0	1	1	0	1	0
1	0	0	0	1	0	0	0	0	0
1	0	0	0	1	1	0	0	0	0
1	0	0	1	0	0	0	0	0	0
1	0	0	1	0	1	0	0	0	0
1	0	0	1	1	0	0	0	0	0
1	0	0	1	1	1	0	0	0	0
1	0	1	0	0	0	1	0	1	0
1	0	1	0	0	1	1	0	1	0
1	0	1	0	1	0	0	0	0	0
1	0	1	0	1	1	0	0	0	0
1	0	1	1	0	0	0	0	0	0
1	0	1	1	0	1	0	0	0	0
1	0	1	1	1	0	0	0	0	0
1	0	1	1	1	1	0	0	0	0
1	1	X	X	X	X	X	X	X	X

Tabla 7: Estados posibles al apretar P2

A partir de la *Tabla 7* se obtienen las ecuaciones del bloque 2 y son las siguientes:

$$D_A = A * \overline{S1} * \overline{S2} + \overline{A} * \overline{B} * P2 * S3$$

$$D_B = B * \overline{S2} * \overline{S3} + \overline{A} * \overline{B} * P2 * S1$$

A partir de las ecuaciones obtenidas se ha diseñado la programación del problema mediante un sistema secuencial, en el cual se han utilizado todas las puertas lógicas necesarias, 4 entradas analógicas, 2 salidas analógicas y dos biestables, uno por cada ecuación. A continuación, se muestra en la *Figura 23* la implementación de este bloque.

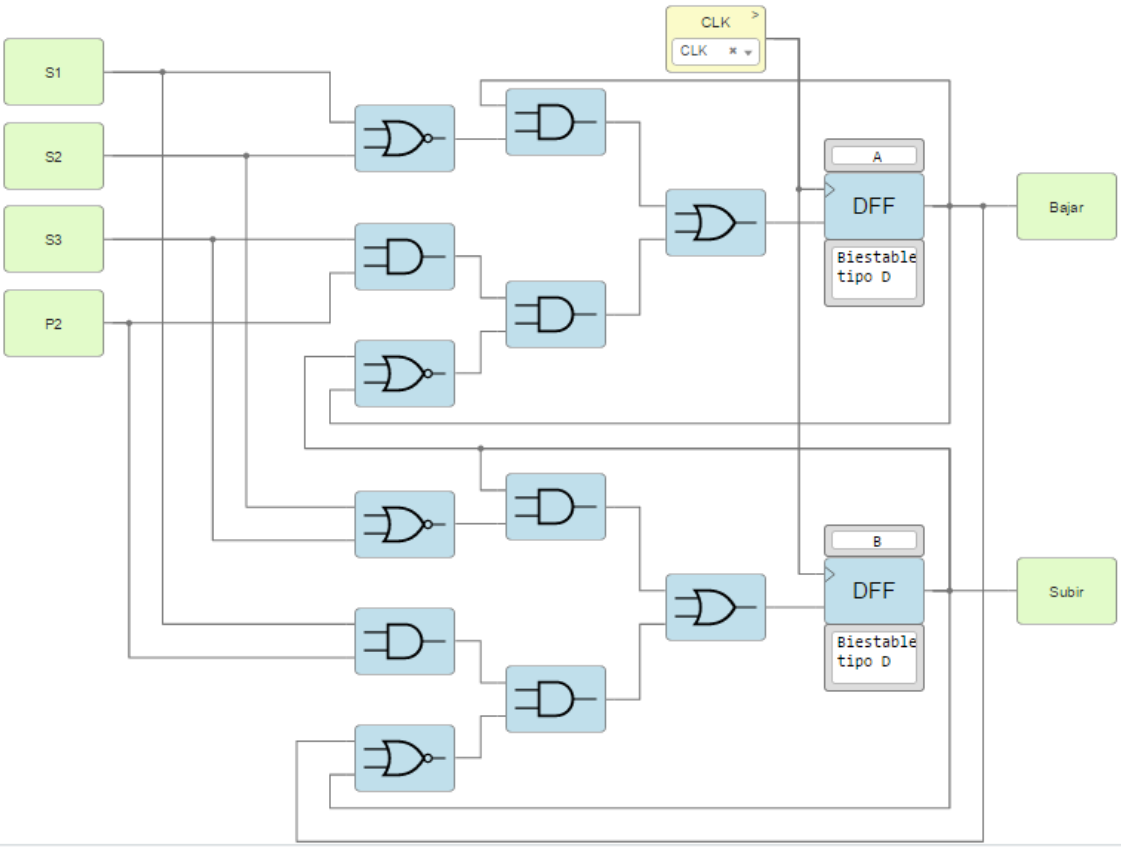


Figura 23: Diseño en Icestudio del funcionamiento apretando P2

5.4.3 Bloque 3 – Funcionamiento al apretar P3

La última parte consiste en cómo funcionará el ascensor cuando se apriete el pulsador P3. Por tanto, estando el ascensor en la primera planta, al apretarse P3 este debe subir a la tercera planta. En esta parte, los únicos estados del ascensor serán *Subir* o *Parado*. En la *Figura 24* se muestra el diagrama de estados del bloque 3.

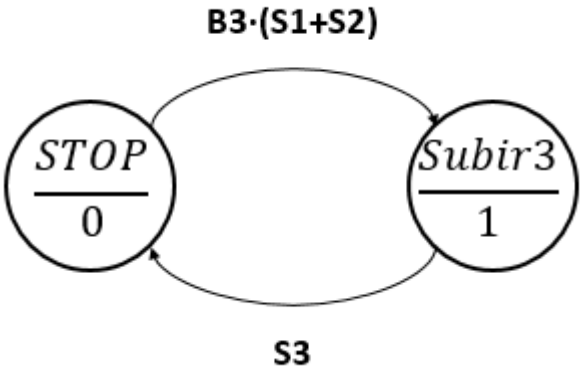


Figura 24: Diagrama de estados P3

Para la resolución de esta parte se necesitará un solo biestable D, ya que se puede determinar los dos estados con un solo bit.

En la tabla siguiente se han expuesto todos los casos posibles en el funcionamiento del ascensor en el momento en que se apriete P3, en función de las entradas activas en ese momento. De esta manera, a partir de la tabla y mediante el método de Karnaugh, se obtendrán las ecuaciones para un funcionamiento correcto.

Estado actual	**INPUTS**				**Estado siguiente**	**Flip-Flop D**
A	P3	S1	S2	S3	An	DA
0	0	0	0	0	0	0
0	0	0	0	1	0	0
0	0	0	1	0	0	0
0	0	0	1	1	0	0
0	0	1	0	0	0	0
0	0	1	0	1	0	0
0	0	1	1	0	0	0
0	0	1	1	1	0	0
0	1	0	0	0	1	1
0	1	0	0	1	0	0
0	1	0	1	0	1	1
0	1	0	1	1	X	X

0	1	1	0	0	1	1
0	1	1	0	1	X	X
0	1	1	1	0	1	1
0	1	1	1	1	X	X
1	0	0	0	0	1	1
1	0	0	0	1	0	0
1	0	0	1	0	1	1
1	0	0	1	1	0	0
1	0	1	0	0	1	1
1	0	1	0	1	0	0
1	0	1	1	0	1	1
1	0	1	1	1	0	0
1	1	0	0	0	1	1
1	1	0	0	1	0	0
1	1	0	1	0	1	1
1	1	0	1	1	0	0
1	1	1	0	0	1	1
1	1	1	0	1	0	0
1	1	1	1	0	1	1
1	1	1	1	1	0	0

Tabla 8: Estados posibles al apretar P3

A partir de la *Tabla 8* se obtienen la ecuación del bloque 3 y es la siguiente:

$$\mathrm{D}_A = (\mathrm{P1} + \mathrm{A}) * \overline{\mathrm{S3}}$$

A partir de la expresión obtenida se ha diseñado la programación del problema implementando un sistema secuencial, este bloque es prácticamente idéntico al bloque 1, excepto por las entradas digitales y la salida digital, las cuales se deben asignar a pines diferentes. A continuación, se puede observar su implementación en Icestudio en la figura siguiente.

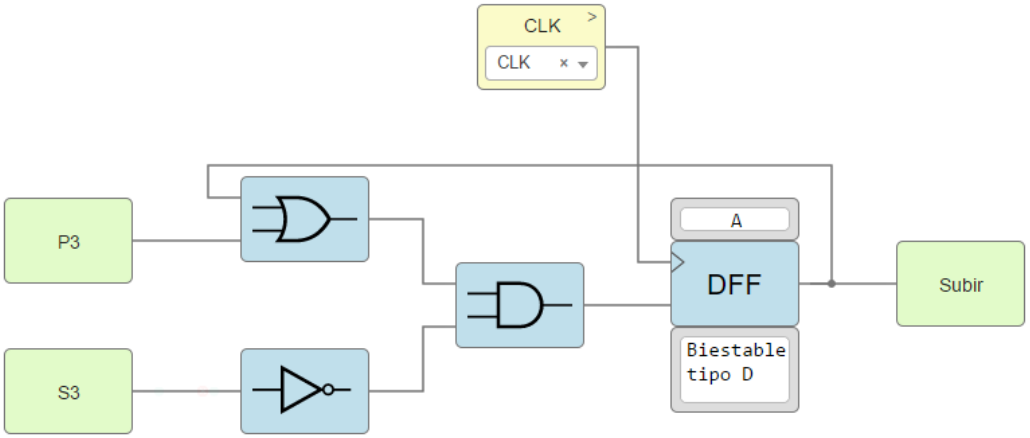


Figura 25: Diseño en Icestudio del funcionamiento apretando P3

5.4.4 Solución final

Por último, la solución final del problema se ha implementado con 6 entradas digitales, 2 salidas digitales y los 3 bloques diseñados anteriormente, todo conectado entre sí de manera que el funcionamiento sea correcto. El diseño final implementado en Icestudio se puede observar en la *Figura 26*.

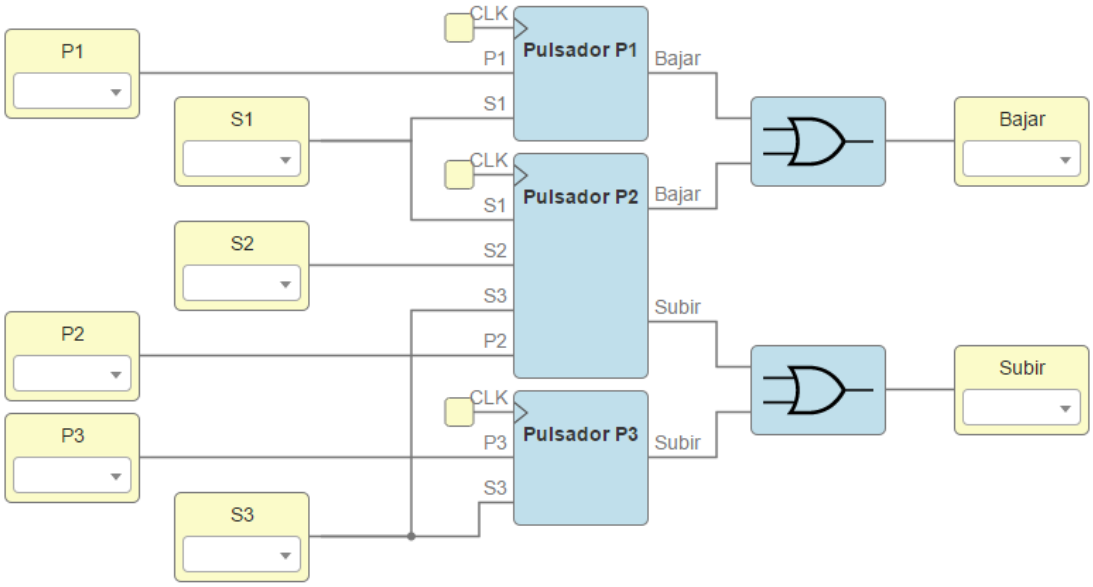


Figura 26: Diseño en Icestudio Problema 3

5.5 Problema 4. Manipulación con cilindros neumáticos

El sistema de la *Figura 27* consiste en un simple proceso industrial de manipulación de piezas basado en cilindros neumáticos. Por tanto, este problema trata de programar el correcto funcionamiento del proceso, mediante circuitos secuenciales.

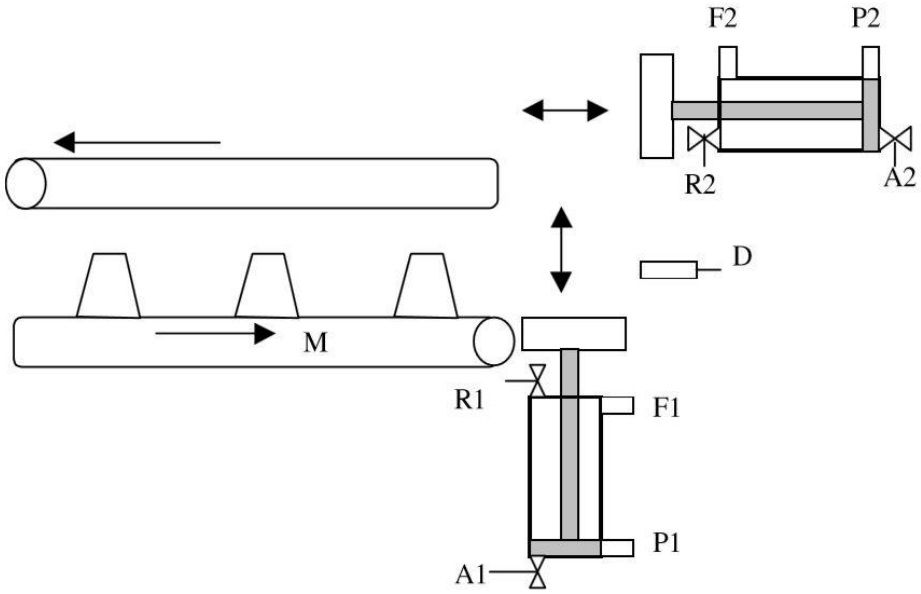


Figura 27: Manipulación con cilindros neumáticos

Por la cinta inferior llegan piezas, esta cinta se mueve mediante la señal **M**. Cuando una pieza se sitúa encima de la superficie de elevación el detector óptico **D**, se activa. En ese momento hay que parar la cinta, subir el cilindro hasta su posición superior (hasta que se activa el detector magnético **F1**), mover el cilindro horizontal hasta **F2** y volver los dos cilindros a su posición inicial (**P1** y **P2** activos). Para hacer avanzar los cilindros se debe activar las señales **A1** y **A2** respectivamente, y para hacerlos retroceder las señales **R1** y **R2**. Hay que tener en cuenta que, si se desactiva cualquiera de las dos señales, el cilindro correspondiente se queda parado. La cinta superior está siempre en marcha (la controla otro proceso).

Se tienen, además, un pulsador de marcha y otro de parada. Cuando se inicia el automatismo, el sistema debe estar en reposo y al pulsar la marcha se pondrá en funcionamiento. Cuando se pulse la parada el sistema acabará de trasladar la última pieza (si hay alguna en la plataforma de subida) antes de volver al inicio y quedar en reposo.

SOLUCIÓN:

Para facilitar la resolución del problema, se ha dividido en tres bloques distintos y se han resuelto uno por uno para que la solución final al problema sea el conjunto de los tres bloques.

5.5.1 Bloque 1 – Funcionamiento del cilindro 1 (Vertical)

A continuación, en la *Figura 28*, se muestra el diagrama de estados de esta primera parte y se puede observar que el cilindro vertical tiene tres estados posibles, *Avanzar* (A1), *Retroceder* (R1) y *Parado* (STOP), por lo que se van a necesitar dos biestables D para diseñar el funcionamiento del cilindro.

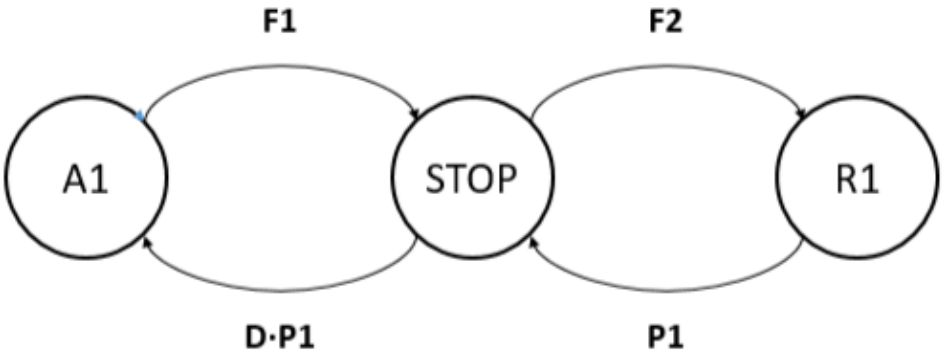


Figura 28: Diagrama de estados del cilindro 1

En la tabla siguiente se han expuesto todos los casos posibles en el funcionamiento del cilindro vertical, en función de los sensores activos en ese momento.

Estado actual		INPUTS				Estado siguiente		Flip-Flop D	
A	B	F1	F2	P1	D	An	Bn	DA	DB
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	1	0	0	0	0
0	0	0	0	1	0	0	0	0	0
0	0	0	0	1	1	0	1	0	1
0	0	0	1	0	0	1	0	1	0
0	0	0	1	0	1	1	0	1	0
0	0	0	1	1	0	0	0	0	0
0	0	0	1	1	1	X	X	X	X
0	0	1	0	0	0	0	0	0	0
0	0	1	0	0	1	0	0	0	0
0	0	1	0	1	0	0	0	0	0
0	0	1	0	1	1	X	X	X	X

0	0	1	1	0	0	1	0	1	0
0	0	1	1	0	1	1	0	1	0
0	0	1	1	1	0	X	X	X	X
0	0	1	1	1	1	X	X	X	X
0	1	0	0	0	0	0	1	0	1
0	1	0	0	0	1	0	1	0	1
0	1	0	0	1	0	0	1	0	1
0	1	0	0	1	1	0	1	0	1
0	1	0	1	0	0	0	1	0	1
0	1	0	1	0	1	0	1	0	1
0	1	0	1	1	0	0	1	0	1
0	1	0	1	1	1	0	1	0	1
0	1	1	0	0	0	0	0	0	0
0	1	1	0	0	1	0	0	0	0
0	1	1	0	1	0	X	X	X	X
0	1	1	0	1	1	X	X	X	X
0	1	1	1	0	0	0	0	0	0
0	1	1	1	0	1	0	0	0	0
0	1	1	1	1	0	X	X	X	X
0	1	1	1	1	1	X	X	X	X
1	0	0	0	0	0	1	0	1	0
1	0	0	0	0	1	1	0	1	0
1	0	0	0	1	0	0	0	0	0
1	0	0	0	1	1	0	0	0	0
1	0	0	1	0	0	1	0	1	0
1	0	0	1	0	1	1	0	1	0
1	0	0	1	1	0	0	0	0	0
1	0	0	1	1	1	0	0	0	0
1	0	1	0	0	0	1	0	1	0
1	0	1	0	0	1	1	0	1	0
1	0	1	0	1	0	X	X	X	X
1	0	1	0	1	1	X	X	X	X
1	0	1	1	0	0	X	X	X	X
1	0	1	1	0	1	X	X	X	X
1	0	1	1	1	0	X	X	X	X
1	0	1	1	1	1	X	X	X	X
1	1	X	X	X	X	X	X	X	X

Tabla 9: Estados posibles del cilindro 1

A partir de la *Tabla 9* y mediante el método de Karnaugh, se obtienen las ecuaciones del bloque 1 y son las siguientes:

$$D_A = A * \overline{P1} + \overline{B} * F2 * \overline{P1}$$

$$D_B = B * \overline{F1} + \overline{A} * P1 * D$$

A partir de las ecuaciones obtenidas se puede diseñar la programación del problema mediante circuitos secuenciales. Para programar esta parte una serie de puertas lógicas, 4 entradas digitales, 2 salidas digitales y, ya que se deben cumplir dos expresiones, dos biestables tipo D. A continuación, se muestra su implementación en Icestudio.

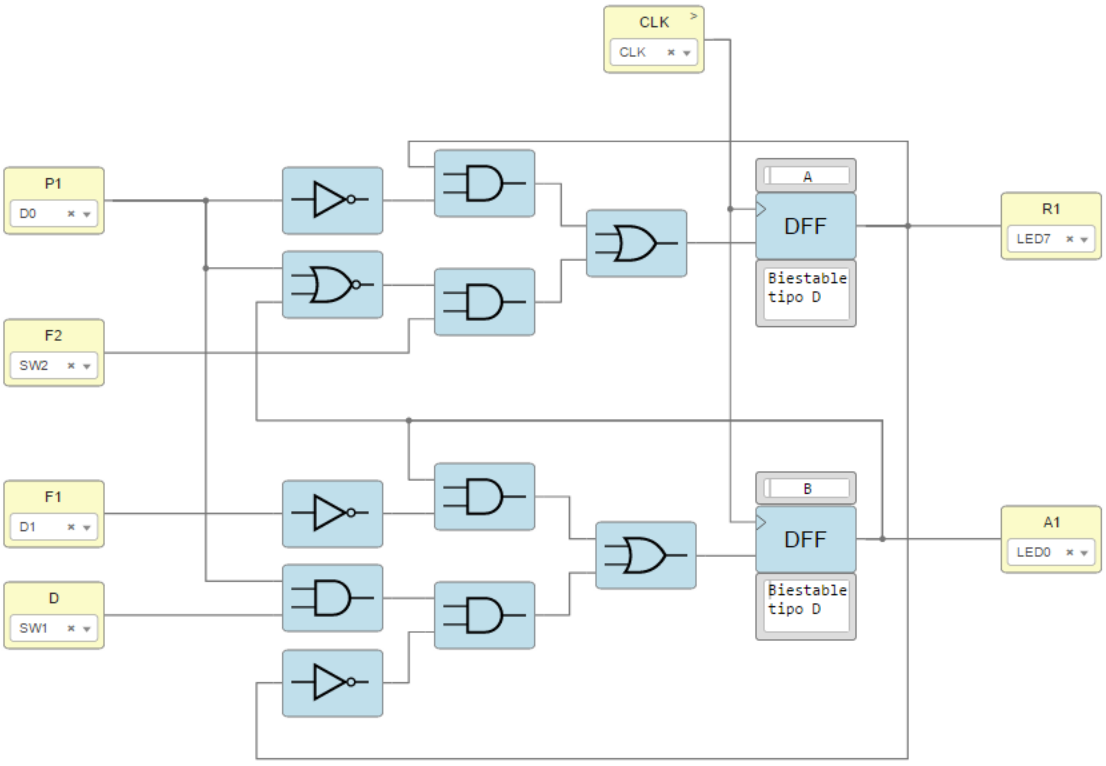


Figura 29: Diseño en Icestudio del funcionamiento del cilindro 1

5.5.2 Bloque 2 – Funcionamiento del cilindro 2 (Horizontal)

Esta segunda parte, correspondiente al funcionamiento del cilindro 2, tiene también 3 estados posibles. Básicamente, es el mismo funcionamiento que el cilindro 1 pero su funcionamiento dependerá de solamente 3 entradas digitales.

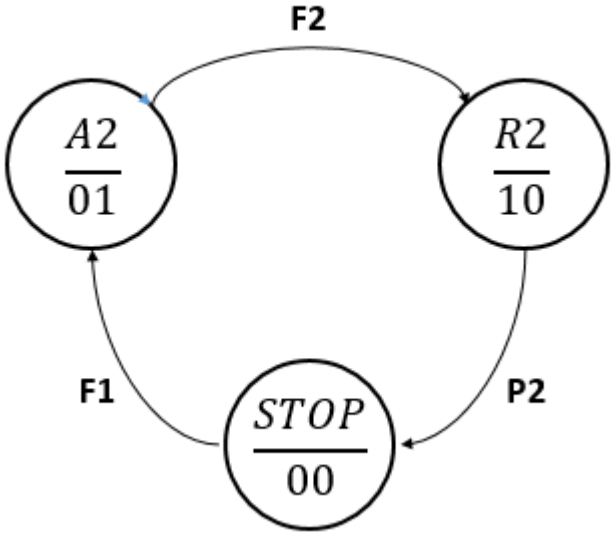


Figura 30: Diagrama de estados del cilindro 2

La *Tabla 10* es la tabla de verdad obtenida del diagrama de estados del cilindro 2. En esta se exponen todos los casos que se pueden dar en el funcionamiento del cilindro 2, en función de las entradas activas.

Estado actual		**INPUTS**			**Estado siguiente**		**Flip-Flop D**	
A	B	F1	F2	P2	An	Bn	DA	DB
0	0	0	0	0	0	0	0	0
0	0	0	0	1	0	0	0	0
0	0	0	1	0	0	0	0	0
0	0	0	1	1	X	X	X	X
0	0	1	0	0	0	1	0	1
0	0	1	0	1	0	1	0	1
0	0	1	1	0	0	1	0	1
0	0	1	1	1	X	X	X	X
0	1	0	0	0	0	1	0	1
0	1	0	0	1	0	1	0	1
0	1	0	1	0	1	0	1	0
0	1	0	1	1	X	X	X	X
0	1	1	0	0	0	1	0	1
0	1	1	0	1	0	1	0	1
0	1	1	1	0	1	0	1	0
0	1	1	1	1	X	X	X	X
1	0	0	0	0	1	0	1	0
1	0	0	0	1	0	0	0	0
1	0	0	1	0	1	0	1	0
1	0	0	1	1	X	X	X	X

1	0	1	0	0	1	0	1	0
1	0	1	0	1	0	0	0	0
1	0	1	1	0	1	0	1	0
1	0	1	1	1	X	X	X	X
1	1	X	X	X	X	X	X	X

Tabla 10: Estados posibles del cilindro 2

A partir de la *Tabla 10* se obtienen las ecuaciones del bloque 2 y son las siguientes:

$$D_A = A * \overline{P2} + B * F2$$

$$D_B = B * \overline{F2} + \overline{A} * \overline{B} * F1$$

De seguido, a partir de las ecuaciones obtenidas, se puede diseñar la programación del problema mediante un sistema secuencial, tal y como se observa en la *Figura 31*.

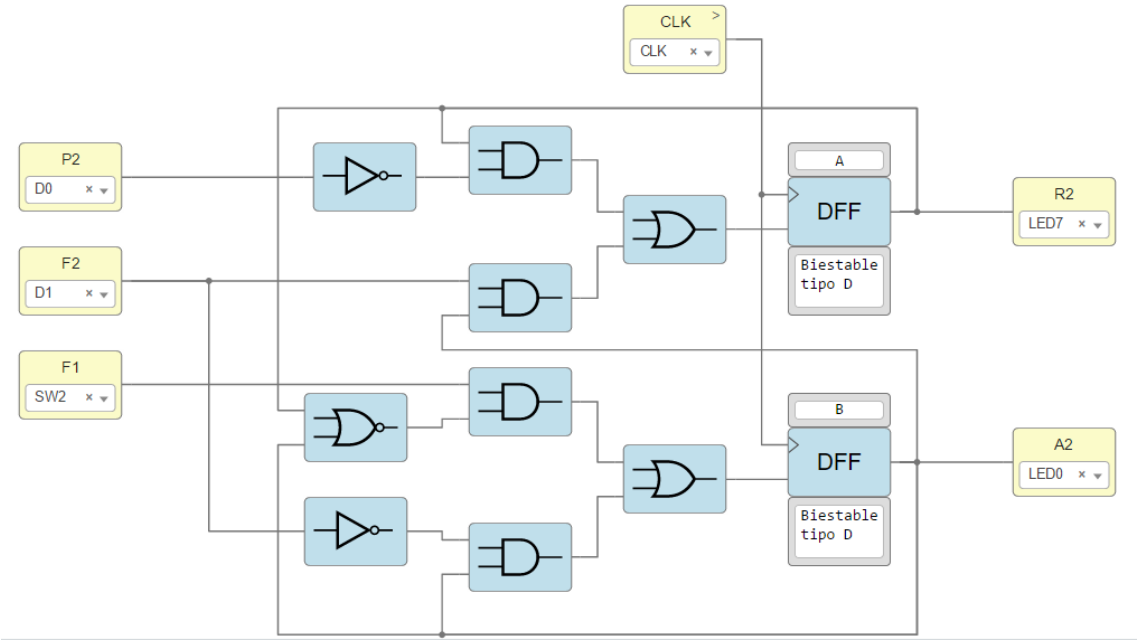


Figura 31: Diseño en Icestudio del funcionamiento del cilindro 2

5.5.3 Bloque 3 – Funcionamiento del motor de la cinta

Por último, el bloque 3 representa el funcionamiento del motor de la cinta transportadora. Esta tendrá que estar en continuo funcionamiento y pararse solamente cuando un objeto este situado sobre el cilindro vertical, pero, además, seguirá parada mientras dicho cilindro vuelva a su posición. Es por eso tiene un estado de "espera" (*Wait*), tal y como se observa en la *Figura 32*.

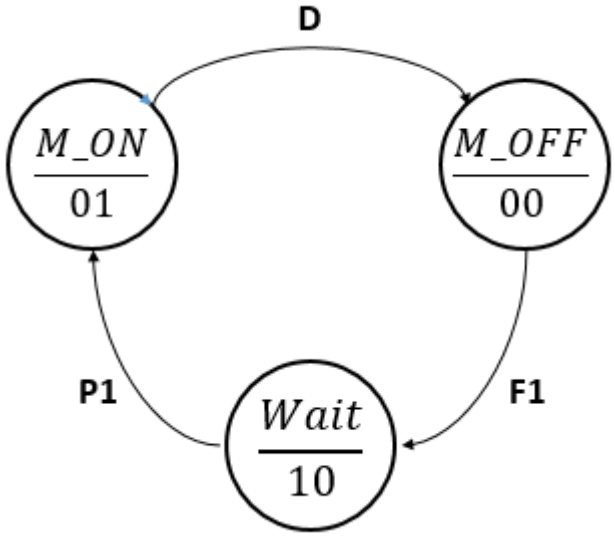


Figura 32: Diagrama de estados del motor de la cinta

Debido que el diagrama anterior tiene 3 estados se necesitan dos bits para representar cada estado en particular y por tanto se utilizarán 2 biestables para conseguir el circuito necesario. Analizando el diagrama de estados, se realiza la tabla de verdad de esta parte del problema, esta tabla se expone a continuación.

Estado actual		INPUTS			Estado siguiente		OUTPUT	Flip-Flop D	
A	B	D	F1	P1	An	Bn	Y	DA	DB
0	0	0	0	0	0	0	0	0	0
0	0	0	0	1	0	0	0	0	0
0	0	0	1	0	1	0	0	1	0
0	0	0	1	1	X	X	0	X	X
0	0	1	0	0	0	0	0	0	0
0	0	1	0	1	0	0	0	0	0
0	0	1	1	0	1	0	0	1	0
0	0	1	1	1	X	X	0	X	X
0	1	0	0	0	0	1	1	0	1
0	1	0	0	1	0	1	1	0	1
0	1	0	1	0	1	0	1	1	0
0	1	0	1	1	X	X	1	X	X

0	1	1	0	0	0	0	1	0	0
0	1	1	0	1	0	0	1	0	0
0	1	1	1	0	0	0	1	0	0
0	1	1	1	1	X	X	1	0	0
1	0	0	0	0	1	0	0	1	0
1	0	0	0	1	0	1	0	0	1
1	0	0	1	0	1	0	0	1	0
1	0	0	1	1	X	X	0	X	X
1	0	1	0	0	1	0	0	1	0
1	0	1	0	1	0	1	0	0	1
1	0	1	1	0	1	0	0	1	0
1	0	1	1	1	X	X	0	X	X
1	1	X	X	X	X	X	X	X	X

Tabla 11: Estados posibles del motor de la cinta

A partir de la *Tabla 11* se obtienen las ecuaciones del bloque 3 y son las siguientes:

$$\mathrm{D}_A = \mathrm{F1} * \overline{(\mathrm{B} * \mathrm{D})} + \mathrm{A} * \overline{\mathrm{P1}}$$

$$\mathrm{D}_B = \mathrm{A} * \mathrm{P1} + \mathrm{B} * \overline{\mathrm{D}} * \overline{\mathrm{F1}}$$

$$\mathrm{Y} = \overline{\mathrm{A}} * B$$

A partir de las expresiones obtenidas se ha diseñado la programación del bloque 3 implementando un sistema secuencial. A diferencia de los bloques anteriores, la resolución de este último bloque está condicionada por tres ecuaciones, las dos primeras, D_A y D_B, correspondientes a los dos biestables y la tercera, *Y*, es una función que depende de las salidas de los biestables. En la Figura 33 se muestra el circuito de este bloque.

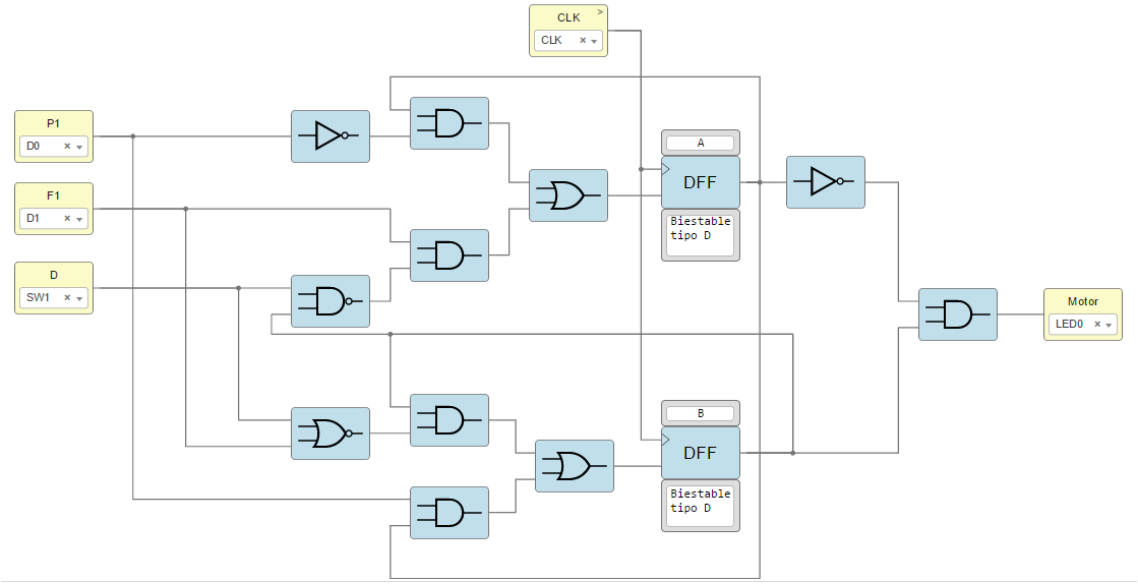


Figura 33: Diseño en Icestudio del funcionamiento de la cinta

5.5.4 Solución final

Finalmente, la solución del problema está compuesta por los tres bloques anteriormente implementados (*Motor*, *Cilindro 1* y *Cilindro 2*) lo que implica la utilización de un total de 6 biestables, 5 entradas digitales y 5 salidas digitales. En la figura siguiente está representado el circuito del sistema secuencial que resuelve este problema.

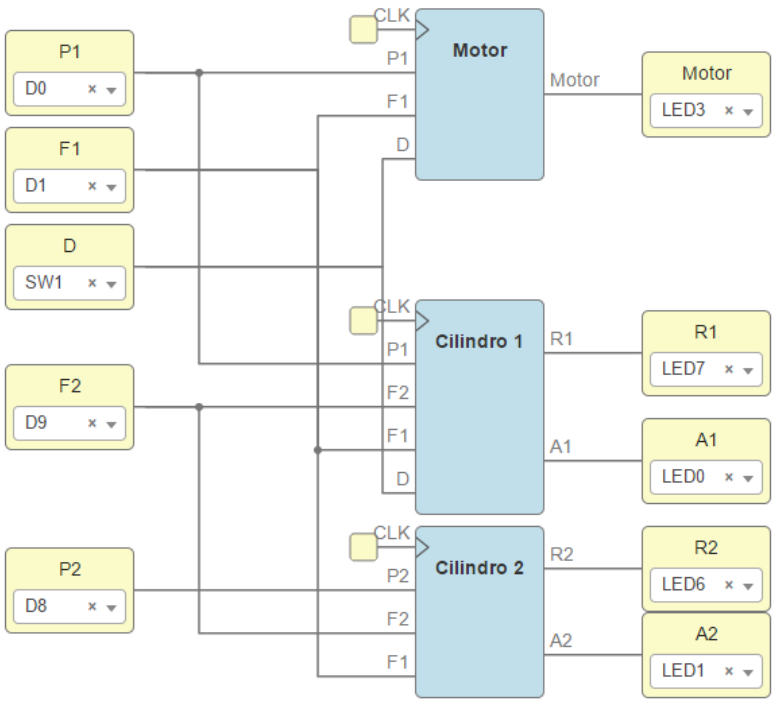


Figura 34: Diseño en Icestudio del Problema 4

5.6 Conclusiones

Se ha cumplido el objetivo de adaptar la placa IceZUM Alhambra para poder utilizarla en prácticas de automatización de sistemas. Esto se ha logrado siguiendo los siguientes pasos:

En primer lugar, se ha investigado sobre el software de código abierto disponible para seguir luego realizando un estudio de mercado sobre las placas FPGA de código abierto o compatibles con el software encontrado, Icestudio. Como conclusión del estudio de mercado, la placa IceZUM Alhambra es la opción que mejor encaja con el objetivo del proyecto.

Seguidamente, se efectuó un segundo estudio de mercado sobre cualquier shield comercial que cumpla las exigencias del proyecto. No se obtuvo ningún resultado satisfactorio y por consiguiente se procedió a diseñar un circuito impreso o PCB acorde con las necesidades planteadas. El diseño de la PCB se desarrolló con el software EAGLE y se fabricó la placa del circuito impreso en el laboratorio de electrónica de la Universidad Jaume I.

Finalmente, se ha planteado una serie de problemas de automatización y se han resuelto implementando circuitos combinacionales o secuenciales, en función de los requisitos de cada problema en particular. Además, los diseños de programación de cada problema se han implementado en el software Icestudio para luego cargarlas en la placa FPGA y de esta manera realizar una simulación para comprobar el correcto funcionamiento del diseño.

Como conclusión final del proyecto se puede decir que los resultados obtenidos con el uso de la placa FPGA IceZUM Alhambra demuestran que la misma puede ser una herramienta valiosa para realizar prácticas de automatización mediante sistemas combinacionales o secuenciales. Además, puede ser usada en prácticas de otras asignaturas en las que se estudien este tipo de sistemas, como por ejemplo Electrónica Digital. Un hecho importante es que todo el hardware y el software es de código libre, lo cual facilita el acceso a la información y al conocimiento.

6. Bibliografía

1. ***FPGAwars:*** Página web del equipo desarrollador del proyecto "FPGAs Libres" donde se puede consultar cualquier información sobre los proyectos IceZUM Alhambra, Icestudio y otros más relacionados con las FPGAs libres. También se pueden consultar los tutoriales para comprender el funcionamiento de las FPGAs y aprender a programarlas.
   http://fpgawars.github.io/

2. ***All About Circuits - Digital:*** Apartado de la página web "All About Circuits" donde se encuentra toda la información necesaria relacionada a la electrónica digital. Además, en la página general se puede encontrar contenido relacionado con la electrónica, como, por ejemplo: información sobre componente digitales, artículos de prensa relacionados con la electrónica y todo tipo de herramientas para aplicaciones electrónicas.
   https://www.allaboutcircuits.com/textbook/digital/

3. ***Karnaugh 32x8:*** Herramienta que, mediante el método de Karnaugh, las tablas de la verdad de hasta 8 variables.
   http://www.32x8.com/

4. ***Autodesk - EAGLE:*** Página web del software Eagle, es un programa de gran ayuda para el diseño de PCBs.
   https://www.autodesk.com/products/eagle/overview

5. ***Youtube – ElectroTuoriales:*** Página web donde poder consultar y aprender sobre el funcionamiento del software Eagle (Versión 8.0) y el diseño de PCBs.
   https://www.youtube.com/watch?v=8Zdl-Ixke0w&list=PLAISa0qvV-Hfsn_kRU6aGub2KcrmG9nUY

7. Anexos

7.1 Guía de software

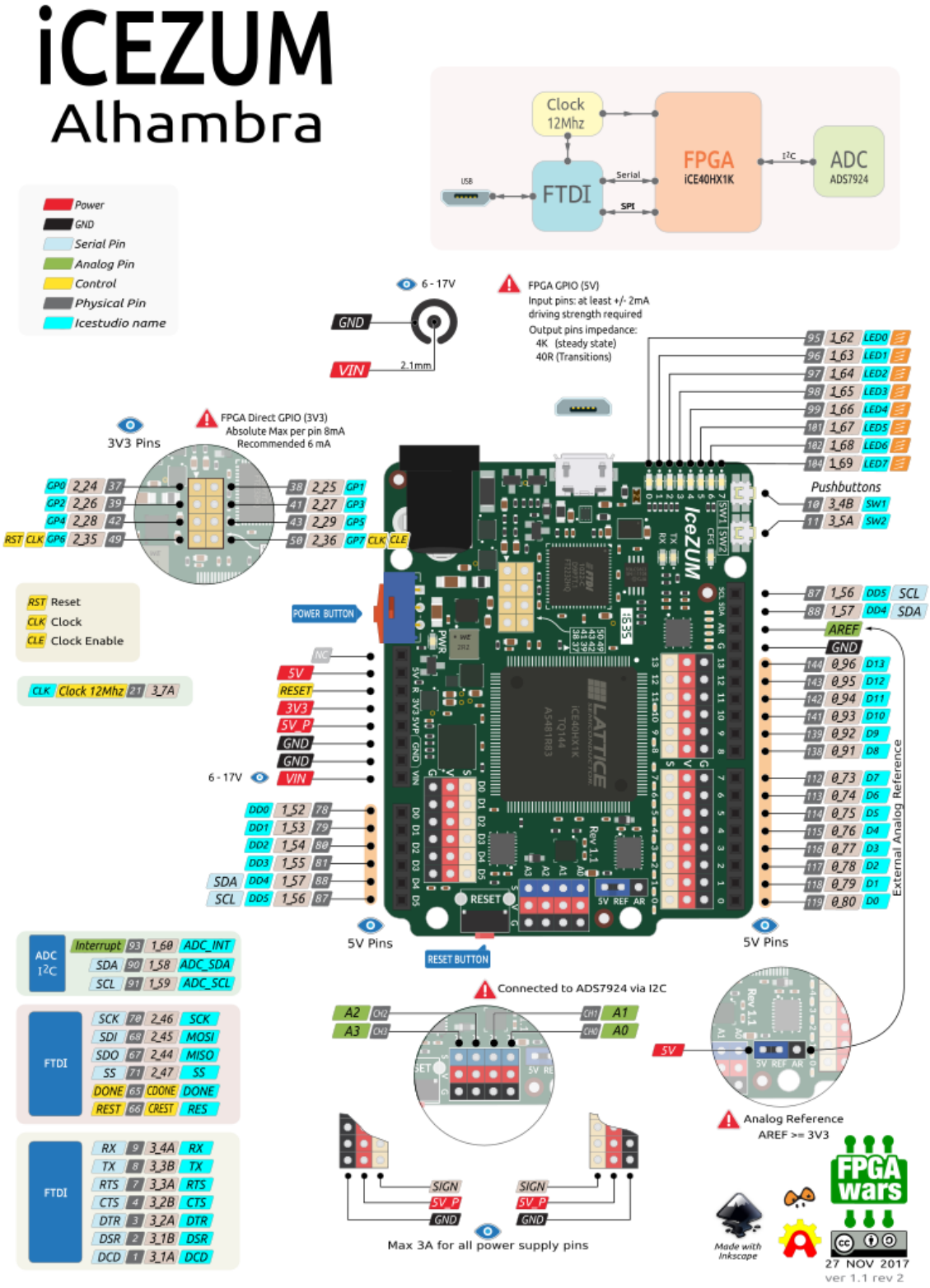
❖ Icestudio

- **Desarrollador:** Jesús Arroyo Torrens
- **Licencia:** Gratuita
- **Sistemas operativos:** Windows, Linux y Mac OS
- **Requisitos del sistema:** No especificado
- **Descarga:** https://github.com/FPGAwars/icestudio
- **Utilizado en este trabajo:** Icestudio 0.3.3 (64 bits)

❖ EAGLE

- **Desarrollador:** Autodesk
- **Licencia:** Gratuita, Estándar y Premium
- **Sistemas operativos:** Windows, Linux y Mac OS
- **Requisitos del sistema:** No especificado
- **Descarga:** https://www.autodesk.com/products/eagle/free-download
- **Utilizado en este trabajo:** EAGLE 9.0.1

7.2 Detalle de los pines de IceZUM Alhambra



7.3 Diagrama electrónico del *shield*